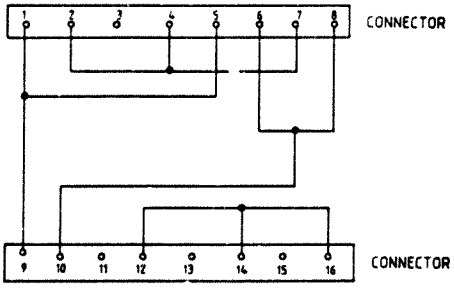


Figure 2.24 Loom interconnection diagram

The success of the short-circuit and open-circuit diagnostic algorithms requires the ordering to be changed to N9:N5:N1. The reason for this reshuffling will be explained shortly. The serial data pattern stored with respect to each new test bit node, will undergo this ordering modification. This ensures storage from the highest to the lowest node number, for each test bit node measurement cycle.

Since each node in the test system will in turn become a test bit node, inherent redundancy is found in the form of measurement data duplication. This is best demonstrated by means of an example.

Table 2.1 Measurement data duplication

MEASUREMENT CYCLE	MODIFIED STORED SHORT-CIRCUIT DATA PATTERN
Test bit corresponding to N1	N9 N5 N1
Test bit corresponding to N5	N9 N5 N1
Test bit corresponding to N9	N9 N5 N1

The redundancy is removed by ensuring that any node, known to be shorted to a previous test bit node, is excluded from the test bit node generation list. Once again, enhancing measurement speed as well as unnecessary software overhead. Table 2.2 describes how the entire interconnection configuration of the loom under test (see Figure 2.24) is acquired. Measurement cycles are depicted in a step-by-step format, focussing all detail on measurement data storage patterns.

Due to the elimination of the inherent redundancy found in this type of digital measuring technique, each connector pin number or node is represented once only. Not only being a concise way of storing measurement data, the entire short and open circuit list needed to reconstruct the wiring diagram of the loom is included. The actual information is contained in the positioning or ordering of the nodes in the array. This approach allows for simple, high speed, minimum code fault detecting algorithms, avoiding the burden of sifting through chunks of useless information.

Table 2.2 Loom interconnection configuration data storage pattern

MEASUREMENT CYCLE	STORED SHORT-CIRCUIT DATA PATTERN	MODIFIED STORED DATA PATTERN
Test bit node N1	N5 : N1 : N9	N9 : N5 : N1
Test bit node N2	N7 : N4 : N2	N7 : N4 : N2
Test bit node N3	N3	*N3
Test bit node N6	N8 : N6 : N10	N10 : N8 : N6
Test bit node N11	N11	N11
Test bit node N12	N16 : N14 : N12	N16 : N14 : N12
Test bit node N13	N13	*N13
Test bit node N15	N15	*N15

Combining the modified stored data patterns in order, from test bit node N1 measurement cycle results downwards, the following array represents the entire loom interconnection diagram.

Interconnection diagram array = N9:N5:N1:N7:N4:N2:N3:N10:N8:N6:N11:N16:N14:N12:N13:N15

* An isolated test bit generation node indicates that no other connector pins are shorted to this particular node. In other words the test bit node is open-circuit with respect to all other nodes.

Fault diagnostic algorithms

The principles behind automated go/no-go testing of looms can be divided into three basic tasks:

- permanent storage of model interconnection configuration;
- loading of model and current wiring diagrams;
- implementation of fault diagnostic algorithms.

Permanent storage of model interconnection configuration. This entails the selection of a loom, wired exactly according the latest documentation from the batch under test. Undergoing the measurement cycle procedure just described, the short and open-circuit data storage pattern is serially transmitted to the MPU. This information is then permanently transferred to an EPROM cartridge. The EPROM basically contains the model wiring diagram of a particular type of loom, which will then be used for referencing purposes. Due to the non-volatile nature of the EPROM, this permanent storage exercise needs to be performed once only. Updates may be required, due to documentation modifications or damage to the EPROM itself.

Loading of model and current wiring diagrams. Both the current as well as the EPROM-based model wiring diagrams of the loom under test are loaded into the MPU. The EPROM data type must correspond to the documentation of the current loom under test.

Implementation of fault diagnostic algorithms. A comparison of the data array representing the loom's current wiring layout with that containing its model interconnection configuration, is made. By means of software techniques, any differences found can be interpreted in terms of short and open-circuit information among the nodes.

For explanation purposes the example loom shown in Figure 2.24 constitutes the model interconnection configuration. Figure 2.25 illustrates a faulty loom of the same type, representing the current wiring configuration. The loom under test is chosen to be faulty, almost beyond recognition, enabling a detailed examination of the fault diagnostics.

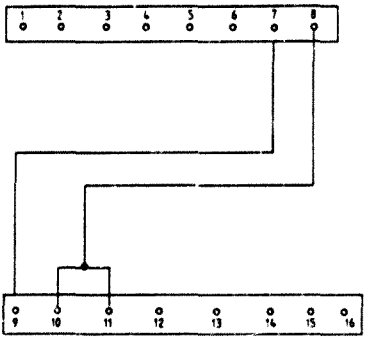


Figure 2.25 Faulty loom interconnection configuration

On completion of the loading task, the data arrays MODEL (I) and CURRENT (I) display the following interconnection information, in the form of strategically positioned node numbers as shown in Table 2.3.

Table 2.3 Interconnection configuration data arrays after loading

MODEL (J) = (N9:N5:N1:N7:N4:N2:N3:N10:N8:N6:N11:N16:N14:N12: N13:N15)
CURRENT (I) = (N1:N2:N3:N4:N5:N6:N9:N7:N11:N10:N8:N12:N13:N14: N15:N16)

Since each node is uniquely represented and the chosen test setup has a maximum capacity of 16 node termination points, indexers I and J extend from 1 through to 16.

Open-circuit detection algorithm. The algorithm is performed according to the following procedure:

```
LOOP FROM I = 1 TO CAPACITY

(* This implies that all events found within the loop will be
   executed 'capacity' times. Capacity is the maximum number
   of loom termination points or nodes that the test
   measurement system can accomodate. In this particular case
   the loop is entered with I = 1, and exits with I = 17 *.)
```

1. Read the node number at CURRENT (I).

2. Find the position J, of this node, in the model array.

3. Find all nodes shorted to the node corresponding to CURRENT (I). This is accomplished by reading all nodes that satisfy the condition CURRENT (I) < CURRENT (I-1).

   (* The relevance of this condition is due to the fact that all nodes found to be shorted to a particular test bit node, are stored with the larger numbers first *.)

4. Find all the nodes shorted to the node corresponding to MODEL (J). This is achieved by reading all nodes satisfying the condition MODEL (J) < MODEL (J-1).

5. Compare the two packs of short-circuit information.

6. Any node found in the short-circuit data pack corresponding to the MODEL array, but not in that of the CURRENT array, is defined as an open-circuit.

7. The fault/s will be stored as an open-circuit between the node at address CURRENT (I) and the missing node(s). This information is placed in the OPEN array.

Repeat loop until I > CAPACITY.

Together with the above procedure the example loom (see Figures 2.24 and 2.25) will be used to demonstrate the first loop iteration. Refer to Table 2.3 for the data content of arrays MODEL and CURRENT.

1. Node = N1.

2. Position J = 3.

3. Short circuit pattern with respect to N1 = N1, i.e. indicating that node N1 is totally isolated from all the other node points.

4. Short circuit pattern with respect to N1 = N9:N5:N1, i.e. indicating that nodes N9 and N5 are shorted to node N1. Note the order in which the data is stored.

5;6. Nodes not found in the CURRENT array short-circuit data pack = N9 and N5.

7. At this point the array OPEN = N1:N5:N1:N9, indicating node pairs N1 to N5 and N1 to N9 to be open-circuit fault conditions.

Refer to Table 2.4, which contains fault diagnostic results collected after fully implementing the open-circuit algorithm. Summing all the data collected in the OPEN array finds:

OPEN = (N1:N5:N1:N9:N2:N4:N2:N7:N4:N7:N5:N9:N6:N8:N6:N10:N12:N14:
N12:N16:N14:N16).

This information indicates open-circuit fault conditions among the following node pairs:

N1 to N5;
N1 to N9;
N2 to N4;
N2 to N7;
N4 to N7;
N5 to N9;
N6 to N8;
N6 to N10;
N12 to N14;
N12 to N16;
N14 to N16.

Table 2.4 Fault diagnostic results collected from the open-circuit detection algorithm

LOOP ITERATION COUNTER I	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
STEP DESCRIPTION																
1 Node corresponding to current (I)	N1	N2	N3	N4	N5	N6	N9	N7	N11	N10	N8	N12	N13	N14	N15	N16
2 Position J	3	6	7	5	2	10	1	4	11	8	9	14	15	13	16	12
3 CURRENT array short-circuit data packet determined with respect to the node read in step 1	N1	N2	N3	N4	N5	N6	N9	N7 N9	N11	N10 N11	N8 N10 N11	N12	N13	N14	N15	N16
4 MODEL array short circuit data packet determined with respect to the node read in step 1	N1 N5 N9	N2 N4 N7	N3	N4 N7	N5 N9	N6 N8 N10	N9	N7	N11	N10	N8 N10	N12 N14 N16	N13	N14 N16	N15	N16
5 & 6 Comparison of data collected in steps 3 and 4. Detection of data found in step 4 but not in step 3.	N5 N9	N4 N7		N7	N9	N8 N10						N14 N16		N16		
7 Open-circuit fault data stored in 'open' array	N1 N5 N1 N9	N2 N4 N2 N7		N4 N7	N5 N9	N6 N8 N6 N10						N12 N14 N12 N16		N14 N16		

The breakdown may be caused by any combination of nodes shown in the OPEN array. The solution would be to physically locate the fault. However, the validity of the fault diagnostic results quickly points to the problem area. By visually comparing the wiring diagrams shown in Figures 2.24 and 2.25, the above information is easily verified.

Short-circuit detection algorithm

This algorithm is performed according to the following procedure:

LOOP FROM I = CAPACITY DOWN TO 1.

(* Reading nodes starting at the end position of the CURRENT array allows only short-circuit node clusters to be sifted out. Basically only those nodes, involved in a short-circuit configuration, will be examined by the algorithm *.)

1. Read the node number at CURRENT (I).
2. If the condition CURRENT (I) < CURRENT (I-1) is true then:

   (a) find the position J, of the node read in step 1, in the MODEL array;
   (b) find all the nodes shorted to node corresponding to CURRENT (I). This is accomplished by reading all nodes satisfying the condition CURRENT (I) < CURRENT (I-1);

(c) find all nodes shorted to the node corresponding to MODEL (J). This is achieved by reading all nodes satisfying the condition
MODEL (J) < MODEL (J-1);

(d) compare the two packs of short-circuit information. Any node found in the short-circuit pack corresponding to the CURRENT array, but not in that of the MODEL array, is defined as a short circuit;

(e) the fault(s) will be stored as a short circuit between the node at address CURRENT (I) and the missing node(s). This information is placed in the SHORT array.

Repeat loop until I = 1.

The example loom (see Figures 2.24 and 2.25) will be used to demonstrate the first loop iteration. Refer to Table 2.3 for the data content of arrays MODEL and CURRENT.

1. Node = N16, i.e. I = 16.
   Since the condition CURRENT (I) < CURRENT (I-1) is false, no fault information is stored in the SHORT array. This completes the first cycle through the loop.

Refer to Table 2.5, displaying fault diagnostic results collected after fully implementing the short-circuit algorithm. Summing all the data collected in the SHORT array finds:

SHORT = (N8:N11:N10:N11:N7:N9)

This information indicates short-circuit fault conditions among the following node pairs:

N8 to N11;
N10 to N11;
N7 to N9.

By visually comparing the wiring diagrams shown in Figures 2.24 and 2.25, this information can easily be verified.

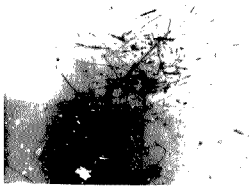


Table 2.6 Fault diagnostic results collected from the short-circuit detection algorithm

LOOP ITERATION COUNTER I		16	15	14	13	12	11	10	9	8	7	6	5	4	3	2
STEP DESCRIPTION																
1	Node corresponding to CURRENT (I)	N16	N15	N14	N13	N12	N8	N10	N11	N7	N9	N6	N5	N4	N3	N2
2	Test condition CURRENT (I) < CURRENT (I-1)	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	TRUE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
2a	Position J						9	8		4						
2b	CURRENT array short-circuit data pack determined with respect to the node read in step 1						N10 N11	N11		N9						
2c	MODEL array short circuit data pack determined with respect to the node read in step 1						N10									
2d	Comparison of data collected in steps 2b and 2c. Detection of data found in 2b but not in 2c						N11	N11		N9						
2e	Short-circuit faults stored in the SHORT array						N8 N11	N10 N11		N7 N9						

Self-Testing

Due to the digital nature of the measurement system, a fully comprehensive self-test can easily be performed on the go/no-go measurement card. From an equipment maintenance point of view, fault-finding times are greatly increased, indirectly providing a cost saving. The operators' confidence in the test system is enhanced by allowing an automatic self-test option, possibly leading to a better man/machine relationship.

The self test comprises two basic phases:

- Scanning of the entire node bus, reading only those nodes representing a logic 'low' level. Instead of generating a test bit node list, each line is passively scanned reading only 'low' level logic states to the CURRENT array. This information represents short-circuit fault conditions generated by the measurement card.

- Scanning of the entire node bus in conjunction with a special self test loom. The wiring configuration of the self-test loom comprises individual shorted wire pairs. Thus any open-circuit condition due to board failure will be detected.

Measurement board failure could arise due to any of the following conditions: integrated circuit failure, faulty interface wiring, or damaged PCB tracks and components.

Assuming the measurement card to has no fault conditions present the node patterns displayed in the CURRENT array will be:

- for phase 1: (short-circuit detection configuration) CURRENT = (NULL ARRAY), implying no short-circuit condition exists;

- for phase 2: (open-circuit detection configuration) CURRENT = (N9:N1:N10:N2:N11:N3:N12:N4:N13:N5:N14:N6:N15: N7:N16:N8), implying the self-test loom configuration. In this phase the data pattern storage mechanism, is the same as for the fault diagnostic algorithms just described.

Note that in any short-circuit cluster the high value node numbers are always stored first, down to the actual test bit node.

Again, the 16-node measurement setup shown in Figure 2.23 is used to explain the details of the self-test concept. The idea would be to store model information gathered as a constant look-up table, against which phase 2 self-test data can be referenced. Taking the results from phase 2 of a fully functional 16-node measurement card, the look-up array will be:

OPEN-S/TEST = (N9:N1:N10:N2:N11:N3:N12:N4:N13:N5:N14:N6:N15: N7:N16:N8).

The example shown assumes the following measurement card fau conditions:

- connector interface wires to node points N1, N3 and N8 are broken off;
- damaged buffer bits corresponding to node lines N7, N10 and N16 are permanently pulled to ground, resulting in a constant board generated short-circuit condition.

Without any external loom connection to the tester, phase 1 is implemented, generating the following pattern:

CURRENT = (N7 : N10 : N16)

This information directly implies short-circuit fault conditions on lines N7, N10 and N16. This exposes the faults in the second part of the example.

Connecting the self-test loom to the system, phase 2 is then implemented generating the following node pattern:

CURRENT = (N1:N16:N15:N10:N7:N2:N3:N12:N4:N13:N5:N14:N6:N8:N9:
N11).

By directly comparing data between the CURRENT and OPEN-S/TEST arrays, nodes N1+N9, N3+N11 and N8+N16 are found to be open-circuit faults. This information directly implies board-generated open-circuit fault conditions on these particular node lines. This exposes the faults in the first part of the example.

Conclusion

The ultimate purpose of any automatic loom tester, covering the go/no-go test philosophy domain, is to detect faults in wiring configurations. The design challenge is obviously to find the most efficient techniques of performing this function, offering the user:

- low equipment cost;
- fast measurement times;
- intelligent fault-finding algorithms.

Due to the digital nature of this measurement system, the following advantages are offered:

- components are inexpensive and readily available;
- hardware real-estate can be kept to a minimum;
- the power of the measurement system is software-based.

Many of the testers available make use of analogue measurement systems. Apart from the accuracy overkill for this type of test philosophy, components are costly (i.e. relays, analogue-to-digital converters, etc.). The use of relays implies wear and tear, which is inconvenient for the user as they eventually have to be replaced.

The go/no-go measurement card module tests 50 nodes in less than one second. This can be extremely advantageous when testing large looms of the order of hundreds of wires, involving many go/no-go measurement modules to be chained together. When testing large looms of considerable volume, slow measurement speeds could prove to be costly.

An anomaly common to fault finding algorithms found in many of the testers [illegible] is shown in the following setup:

node 1 [illegible] short to node 2; multiple faults consist of a known [illegible] is actually open, and an unknown short between node [illegible] node 3.

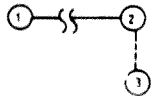


Figure 2.26 Multiple faults with a known short circuit

The results were: during the first run, the test picked up that node 1 was open to node 2. The test did not pick up that 2 was shorted to 3 until the open error was corrected.

During the second run, the short circuit between 2 and 3 was detected.

If the test and maintenance operations are any distance apart, running forwards and backwards could delay a product, resulting in a lower production yield. One of the major advantages of the fault finding diagnostics discussed in the measurement software section, is that the entire fault list of the unit under test is fully displayed on the first run.

3 TECHNICAL DEVELOPMENT AND IMPLEMENTATION OF THE POWER-LINE TEST PHILOSOPHY

All wires measured in the go/no-go test domain constitute control, status and information lines. These types of signals feed into high impedance inputs, implying a small current handling anything below a couple of milli-ohms. A continuity condition implies resistances ranging from tens of milli-ohms to tens of ohms, depending on the characteristics of the wire defined by the following equation:

$$R = \frac{pL}{A} \qquad \text{equation (1)}$$

R = resistance;
p = resistivity of the conducting material;
L = length of wire;
A = cross-sectional area of wire.

Due to the low current carrying capacity of 'go/no-go' type wires, an accurate determination of resistance has no significant meaning to the measurement results. The basic aim is to determine whether a condition is continuous or discontinuous.

Situations will however arise where it will be of the utmost importance to accurately determine the resistance in a loom wire. This type of testing falls under the POWER-LINE test philosophy. Any line carrying some significant current falls into this category. Although the approximate figure of these currents starts from the milli-ohm range upwards, values of significance on the low side of the scale will vary according to different design constraints. Figure 3.1 illustrates a typical example requiring the need for this area of loom testing.

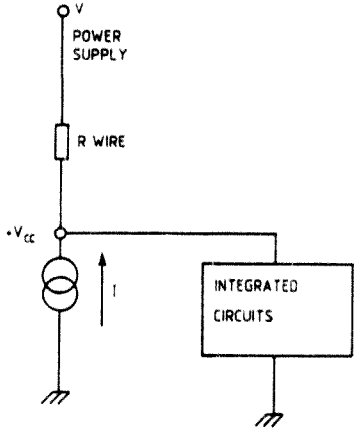


Figure 3.1 Power-line setup

Due to a poor crimping joint between the wire and connector pin or corrosion in the conducting material, the resistance in the power line wire increases from 40 milli-ohms to 2 ohms. Assuming that the current source requirement I = 1 A and the power supply voltage is 5 V, a design constraint in this type of configuration may specify that the PCB supply rail +Vcc, may not drop below 4 V. If this condition occurs, various integrated circuits vital to the functioning of the system, will fail to operate correctly.

Case 1 R_{WIRE} = 40 milli-ohms
Vcc = V power supply - I R_{WIRE}
= 5 - 1 x 0,04
= 4,96 V (acceptable)

Case 2 R_{WIRE} = 2 ohms
Vcc = V power supply - I R_{WIRE}
= 5 - 1 x 2
= 3 V (unacceptable).

3.1 Quality Measurement Card Design

One of the functional specifications mentioned in the automatic loom tester specification was that a facility for accurately measuring the resistance in twenty or more individual loom wires be provided. The QUALITY measurement card provides this very function, allowing the determination of resistance, down to tens of milli-ohms, in twenty different lines.

Keeping in step with the open-endedness and modularity of the design approach, the QUALITY measurement module is also housed on a standard 100 mm x 160 mm Euro-card. These modules are easily expanded by simply daisy-chaining any number of these measurement cards together, in much the same way as described for the go/no-go measurement card.

Figure 3.2 illustrates the basic concept behind the measurement implementation of the QUALITY card. Once the loom under test is connected to the card interface a specific wire can be selected by choosing the appropriate channel in switching bank multiplexer 1. The current source action then pumps a constant current I through the chosen wire of resistance R. The voltage developed across the loom wire,

$$V = IR \qquad \text{equation (2)}$$

is switched to the signal conditioning circuitry via switching bank multiplexer 2. Correct channel selection of both the switching multiplexer banks are controlled by microprocessor driven switching bank controllers.

The signal conditioning circuitry maximizes the amplitude of the input signal, making full use of the Analogue-to-Digital Converter (ADC) conversion range. This effectively allows the detection of minute voltages, limited only by the rated resolution of the ADC. Although such a gain stage introduces various errors into the measurement result, these can be filtered out by software manipulation. This technique once again demonstrates the power and flexibility of a software-based design system.

The ADC converts the analogue inp' voltage, indirectly representing wire resistance, into a digital output in the form of an 8-bit byte. This information is fed into the microprocessor, via the data bus, where the measured data is interpreted, then printed out in the form of a resistance value. Detail concerning software algorithms will be discussed further on.

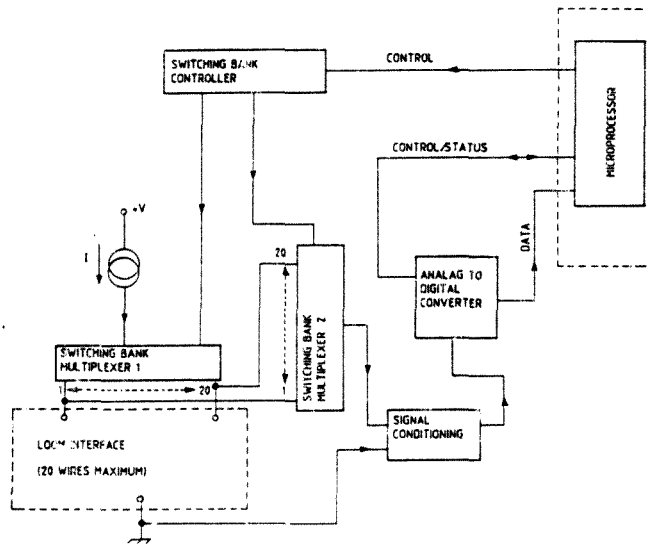


Figure 3.2 Hardware architecture of the quality measurement card

3.1.1 The current source

What influence does current source magnitude have on measurements in this type of testing environment? Many people argue the importance of simulating the actual operating current conditions in a particular wire. A common reason being that a power line exhibiting excessive severed strands will most probably be forced open-circuit when simulated in its 'real-world' environment. This premature fault detection in a test setup usually involves high currents, of the order of a couple of amps upwards. Higher currents and variable current sources are synonymous with high cost implications.

The goal of POWER LINE testing is to detect a degradation in wire resistance. Depending on the sensitivity of the measurement system, the magnitude of the current source I, should have no significant contribution. What effect will the use of a small current source I, say tens of milli-amps, have on a case of severed strands? According to equation (1), the resistance of the wire should increase sufficiently to warrant some attention, and thus being repaired. Although both conditions display the same end result, the use of smaller currents allow for a digital oriented design approach. This introduces low cost and reliability factors, eliminating the need for heavy duty transformers, mechanical relays and large power supplies.

Refer to Figure 3.3 for the basic current source configuration used. The solid-state switching multiplexer bank used, allows a maximum current of 25 milli-amps through each channel. According to this constraint the constant current I was chosen to be 20 milli-amps. The feedback principle of the operational amplifier, implies that the voltage level present on the inverting leg will track that of the non-inverting leg, i.e. the reference voltage.

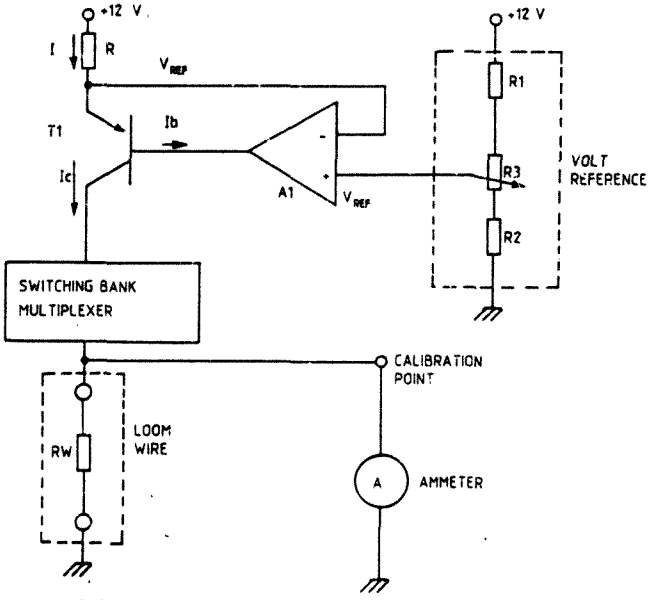


Figure 3.3 Current source

Thus,

$$I = \frac{12 - V_{REF}}{R} \qquad \text{equation (3)}$$

choosing $V_{REF} = 10$ V

the current determination resistor $R = \frac{2}{0,02} = 100$ ohms.

A resistor of this value thus ensures the sourcing of a constant current I = 20 mA.

The resolution of the current source is determined by the tolerances of the voltage reference. By minimizing the voltage span across the potentiometer and choosing a decent number of turns, high precision is ensured. Examine Figure 3.4 for the voltage reference precision analysis which follows.

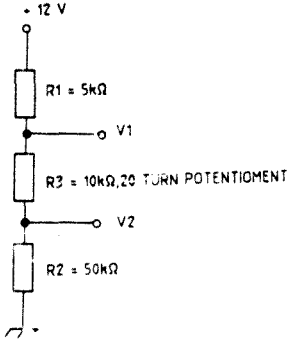


Figure 3.4 Precision voltage reference

$$V1 = \frac{12\ (R2 + R3)}{R1 + R2 + R3} \qquad \text{equation (4)}$$

$$V2 = \frac{12\ R2}{R1 + R2 + R3} \qquad \text{equation (5)}$$

the voltage span across the potentiometer R3.

$$V1 - V2 = 11{,}08\ V - 9{,}23 = 1{,}85\ V.$$

This implies that one turn of the wiper increments the reference voltage by 1,85 V per 20 turns which equals 92 mV. Similarly the half-turn resolution of the reference equals 46 mV. Using a conservative half-turn resolution as the tolerance standard, the current source can be calibrated to within $\pm$ 0,46 mA, i.e. 46 mV/100 ohms of the desired 20 mA current source value. The current source can then safely be specified as having an accuracy to within 2%.

Basing this analysis on a voltage reference tolerance below a half-turn, greater current source accuracy can be achieved. With a high precision ammeter, accuracies of less than 1% can be attained. Transistor T1 introduces a current error by the loss of base current Ib sourced to the op-amp A1.

This base current constitutes a fraction of the constant current I. Thus the actual current flowing through the loom wire Ic equals I - Ib, equation (6). This error can be compensated for by adjusting the voltage reference. During a calibration procedure, all loom wire loads must be disconnected; the only load being an ammeter.

Under no-load conditions the transistor base current saturates to the output short-circuit current of the op-amp. If larger source currents are applied, this prevents destruction of the transistor due to excessive base current drainage. As a result of heating characteristics, the transistor should be derated by at least five times the required current capacity, i.e. 100 mA.

3.1.2 Switching bank multiplexers

Multiplexer banks 1 and 2 consisting of quad digital bilateral switch packages, switch selected wires to the current source and the measurement circuitry.

This base current constitutes a fraction of the constant current I. Thus the actual current flowing through the loom wire Ic equals I - Ib, equation (6). This error can be compensated for by adjusting the voltage reference. During a calibration procedure, all loom wire loads must be disconnected; the only load being an ammeter.

Under no-load conditions the transistor base current saturates to the output short-circuit current of the op-amp. If larger source currents are applied, this prevents destruction of the transistor due to excessive base current drainage. As a result of heating characteristics, the transistor should be derated by at least five times the required current capacity, i.e. 100 mA.

3.1.2 Switching bank multiplexers

Multiplexer banks 1 and 2 consisting of quad digital bilateral switch packages, switch selected wires to the current source and the measurement circuitry.

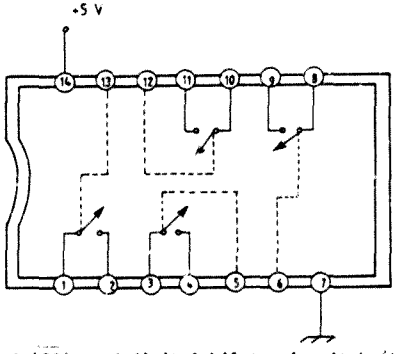


Figure 3.5 A 4066 quad digital bilateral switch (top view)

All four switches may be used separately or in combination. On any single switch, when the control voltage equals ground, the switch remains off and behaves as a very high impedance. When the control voltage equals +5 V, the switch turns on and behaves as a nearly linear, bilateral, 90-ohm resistor. Maximum switching frequency is 5 MHz at 5 V. Dissipation should be kept under 100 mW total, implying that not more than 25 mA can be routed through the circuit.

A problem which may occur, however, is one where the digital switch can give extra output current that is derived from the positive supply, for very low values of load resistance. This is caused by forward biasing of a subrate in transistors internal to the package. Table 3.1 records the results of pumping various currents into a particular digital switch, while monitoring the output current.

Table 3.1 Results of the internal current generation behaviour within the chip

INPUT CURRENT mA	OUTPUT CURRENT mA
20,0	41
15,0	30
12,5	23
8,0	9
7,0	7

Thus, input currents greater than 7 mA trigger this internal chip current generation effect. One way to get around this problem is to call the following pin inputs when delivering current from an outside source to the output load:

pins 2, 3, 9 and [illegible]

This effect applies only to those switches coupling the load (loom wire) to the current source. Switches associated with the high impedance measurement circuitry inputs are required to carry minute currents, making them truly bilateral.

The digital switches forming part of the current source loop, were found to have on-resistances ranging between 80 and 150 ohms. These variations are small enough to be absorbed by the variable resistance effect of transistor T1. Thus, the added resistance of the switching bank multiplexer has no effect on the current sourcing action. A faulty switch on the other hand, displaying a large on-resistance, will transform the current source into a potential divider. Such a condition will result in erroneous measurement data. The switching bank controllers comprise microprocessor driven serial-to-parallel shift registers. These will be discussed further on. As described in the GO/NO-GO measurement card, this form of control technique allows for easy inter-modular expansion.

3.1.3 Signal conditioning circuitry

The measurement limits of the *QUALITY* card will range from approximately 40 milli-ohms to 10 ohms. Depending on the length of the wire, the cross-sectional area, the type of conducting material and the current-carrying requirements, this power-line resistance spectrum covers most areas of interest. The conditioning circuitry ensures that the optimum analogue signal is always fed into the ADC, by supplying a gain stage. This optimization with respect to the full scale input voltage span of the ADC enhances measurement accuracy. A gain factor can be determined by examining the broad specifications of the ADC, as well as inherent errors generated by a gain stage.

The ADC chosen has an 8-bit resolution with a 0 to 5 V full scale input voltage span. The resolution (R) of the ADC is important since the system resolution is determined by this parameter. It is usually defined as the analogue voltage needed to produce a change in the least significant bit of the ADC. For an n-bit ADC, the resolution may be expressed as:

$$R(\text{volts}) = \frac{\text{full scale input voltage}}{(2^n - 1)} \qquad \text{equation (6)}$$

$$= \frac{5}{2^8 - 1} = 20 \text{ mV}$$

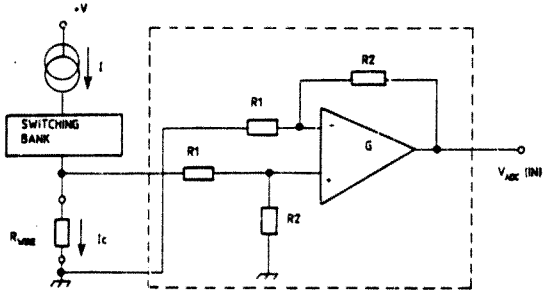


Figure 3.6 Signal conditioning circuitry

Referring to Figure 3.6, the following equation can be derived:

$$V_{ADC}(IN) = G\,[Ic\ R_{WIRE}] \qquad \text{equation (7)}$$

where G is the gain factor.

Choosing a gain G = 25, i.e. R1 = 10 kilo-ohms and R2 = 250 kilo-ohms the absolute conditioned voltage outputs, corresponding to the resistance spectrum specification of the QUALITY card, can be determined.

For $R_{WIRE\ (MIN)}$ = 40 milli-ohms

$V_{ADC\ (IN)}$ = 25 [0,02 . 0,04]

= 20 mV

For $R_{WIRE\ (MAX)}$ = 9 ohms

$V_{ADC\ (IN)}$ = 4,5 V.

The gain is chosen so that the resolution of the ADC 20 mV equals 40 milli-ohms, implying a spectrum which increments in steps of 40 milli-ohms. The 0,5 Volt buffer, i.e. the 9 ohm upper-limit corresponding to 4,5 V and not 5 V, ensures absolute measurement certainty in this region. This allowance takes the following factors into account:

a. Uncertainty concerning the actual full-scale point, i.e. the input analogue voltage corresponding to a digital output change from 11111110B to 11111111B. This point is usually found at Vcc - 1/2 LSB, 5 - 10 mV = 4,9 V. Eliminating the need for this type of verification, ensures high measurement reliability and does not require calibration.

b. Errors generated by the gain stage due to the internal characteristics of an operational amplifier as well as tolerances of external biasing resistors.

Ignoring the above errors would only cause measurement corruption of wire resistance values on the upper end of the spectrum, effectively narrowing the spectrum bandwidth sensitivity.

3.1.4 Noise

Noise conditions and solutions in the ADC and gain stage circuitry can be analysed. To get more perspective on the areas of relevance, the actual components to be chosen will be analysed. Operational amplifiers are available with simple Bi-polar Junction Transistor (BJT), or with Field Effect Transistor (FET), inputs. In general, BJT inputs have large bias currents (approximately 200 nA), small offset voltages (approximately 1 mV), and modest (approximately 2 mega-ohms) input impedances. FET inputs have very small bias currents (approximately 20 pA), large offset voltages (approximately 10 mV), and high input impedances (approximately 100 mega-ohms). The choice of a BJT or FET input Operational Amplifier (OA), depends on whether bias currents or offset voltages are more significant in a particular application.

An OA with a BJT input is preferred in DC applications where low source impedances exist and small offset voltages are important. The QUALITY card current source displays a resistance R_s, approximately 600 ohms, as shown in Figure 3.7. The loom wire load, with resistance values ranging from tens of milli-ohms to a couple of ohms, represents load voltages anywhere between hundreds of micro-volts and hundreds of mV.

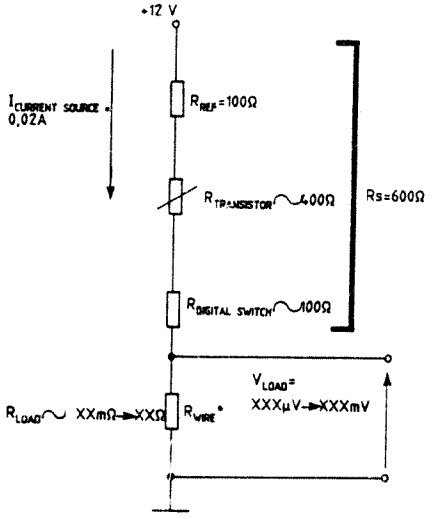


Figure 3.7 Source impedance and load voltage model of the QUALITY card

A small offset voltage, Vos, is important in direct coupled circuits involving low-level DC signals, because Vos is easily amplified along with the signal. This can result in a large, temperature-dependent DC error voltage. The differential front end of an OA consists of a matched pair of BJTs. Because the input transistors of real OAs are not identical, their base currents and base/emitter voltages are unequal. The resulting input offset current, Ios, and input offset voltage, Vos, lead to a net non-zero output voltage even when both inputs are grounded.

Although the presence of Ios and Vos in real OAs leads to an output error voltage, a more serious problem is that these parameters are temperature-dependent, so that this output error voltage drifts with temperature. The following analysis evaluates the maximum lumped error output of the gain stage circuitry, considering both the OA and its external passive components. The temperature range under consideration is between -25 °C and +75 °C, ΔT = 50 °C, with an input resistance R1, of approximately 250 kilo-ohms. Table 3.2 lists the DC error parameters for the chosen LM158A single-supply low-power dual OA.

Table 3.2 DC error characteristics for the LM158A OA

Input offset current	Ios	10 nA (max) at 25 °C
Input offset current drift	ΔIos/ΔT	200 pA/°C (max)
Input offset voltage	Vos	2 mV (max) at 25 °C
Input offset voltage drift	ΔVos/ΔT	15 μV/°C (max)

[TOTAL LUMPED ERROR] MAX

$$= G\left[\frac{\Delta Ios}{\Delta T} R_{IN} + Ios\, R_{IN} + Vos + \frac{\Delta Vos}{\Delta T}\right] \qquad \text{equation (8)}$$

$$= G[2,5\ mV + 2,5\ mV + 2\ mV + 0,75\ mV]$$

$$= G[7,75]\ mV$$

Working with 1% toleranced resistors, implying a maximum gain factor error of approximately 2%, the

TOTAL MAXIMUM LUMPED ERROR

= [25 + 25 x 0,02] 7,75 mV
approximately 200 mV.

Thus the 500 mV voltage error buffer is more than sufficient to absorb the maximum gain stage error.

In any system, a common bus line, from which all voltages are measured, exists. This is referred to as the system common. The analogue measurement subsystem (Figure 3.8) is made up of a number of functional blocks, each with its own common rail, which are in turn connected to the system common. Ideally, all commons should have zero impedance between them, and to ground, but in practice this is not the case. Wires, PCB tracks, and metal chasses all have finite resistances, and some capacitive or inductive coupling always exists between nominally common points and the system ground.

A system ground is established by connecting the system common to earth. It is therefore most important to determine where the return currents from the various functional blocks are flowing. This is because, when the return currents are combined with the finite resistances of their common rails, they generate different potentials along the system common. If two or more adjacent commons are connected, then a ground loop is formed, and the different potentials cause a circulating current to flow.

In addition, the topology of the ground loop can cause it to act as an antenna so that the RF interference that it picks up can contribute to the ground loop current. Figure 3.8 shows how these return currents can cause coupling between the functional blocks in the analogue measurement subsystem. R1, R2 and R3 represent the resistance of the connecting wire; a few inches of 20 gauge wire would typically have a resistance between 1 and 10 milli-ohms. The return currents are normally such that those from the later stages are much greater than those from the early stages (I_1, $<I_2$, $<I_3$).

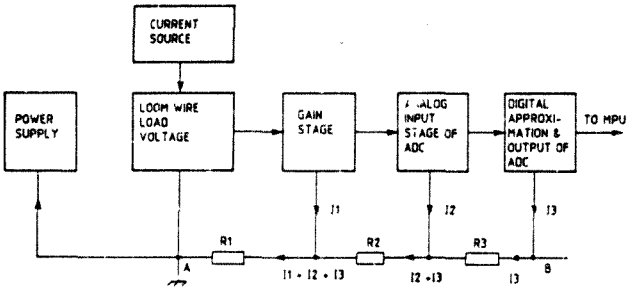


Figure 3.8 Coupling caused by the use of a common path for return currents

This means that the "earth" voltage at point B will be different from that at point A. This may not be much of a problem at point B, since we are dealing with high level signals at this end of the analogue input subsystem. But the fact that I_3 also has an effect on the "earth" of the loom wire load voltage, where the signals are smaller, can lead to significant coupling and consequent corruption of the signal. From an ADC point of view, analogue and digital signal grounds should be kept separate where possible.

This prevents noisy digital signals from flowing into the analogue ground circuit and inducing spurious analogue signal noise. The elimination of this coupling then requires two things. Not only should a single ground be used, but also each functional block should have a separate lead to connect it to ground, as shown in Figure 3.9, so that no two signals share the same return path.

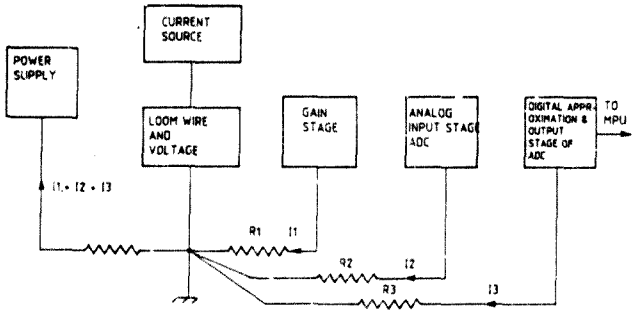


Figure 3.9 Decoupling by the use of separate paths for return currents

3.1.5 Automatic calibration

In order to allow the final software manipulation to accurately convert the ADC digital bytes into actual wire resistance measurement results, the inherent errors found in the gain stage must be calculated. This is especially critical for small wire resistance values, where the signal to noise ratio tends to unity. As previously mentioned, these errors constitute:

a. temperature dependent offset voltages and currents, producing a nett output error voltage, V_{OFFSET};

b. error due to passive component tolerance, resulting in a deviation in the gain from the ideal value G.

Unpredictable variables, such as a continuously drifting temperature environment and differences between like OAs (components), make manual calibration meaningless, if not impossible.

The software control base of the QUALITY card, allows for the easy implementation of auto-calibration techniques. The accurate determination of V_{OFFSET} and G is made possible by the hardware setup, shown in Figure 3.10.

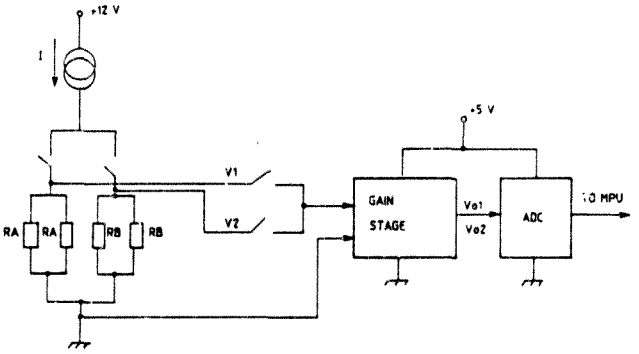


Figure 3.10 Automatic calibration hardware configuration

By alternately switching the R_A and R_B resistor loads, the following equations are derived:

$$V_{O1} = V'_{OFFSET} + G'V_1 \qquad \text{equation (8)}$$

where: G' is the latest gain factor value, and

V'_{OFFSET} is the latest lumped OA output voltage error.

$$V_{O2} = V'_{OFFSET} + G'V_2 \qquad \text{equation (9)}$$

Thus the unknowns and V'_{OFFSET} are calculated as follows:

$$G' = \frac{V_{02} - V_{01}}{V_2 - V_1} \qquad \text{equation (10)}$$

$$V'_{OFFSET} = V_{01} - G'V_1 \qquad \text{equation (11)}$$

Using 1% tolerance resistors, the lumped tolerance in each of the resistive loads,

$$T_Q = \frac{1}{R1 + R2} \sqrt{(R2\ T_{R1})^2 + (R1\ T_{R2})^2} \qquad \text{equation (12)}$$

where $Q = \frac{R1\ R2}{R1 + R2}$ equation (13)

In both cases, i.e. R1 = R2 = RA and R1 = R2 = RB, the calculated lumped tolerance T_Q is approximately 0,7%. The approximate 2,7% error of uncertainty found in this auto-calibration setup, a combination of resistor and current source tolerances, is small enough to be filtered out by the resolution of the ADC. Choosing RA = 1 ohm gives a resistive load equal to 0,5 ohms yielding an ADC input voltage of approximately 25 x 0,02 x 0,5 which equals 0,25 V; the 2,7% error is 7 mV. Similarly choosing RB = 2 ohms gives a 2,7% error in the ADC input voltage which equals 14 mV. The area of uncertainty in the automatic calibration technique is approximately the $\frac{\text{ADC resolution}}{G} \pm 0{,}8$ mV.

having no implications on the measurement system as a whole.

3.1.6 A two-wire quality card example

A two-wire loom capacity QUALITY card example, (Figure 3.11), is used to demonstrate the detailed functioning of the measurement card, from both a hardware and software point of view. Using the specified twenty wire configuration would only produce the same results, merely duplicating the basic operating principles. The following discussion refers to the circuit diagram illustrated in Figure 3.11.

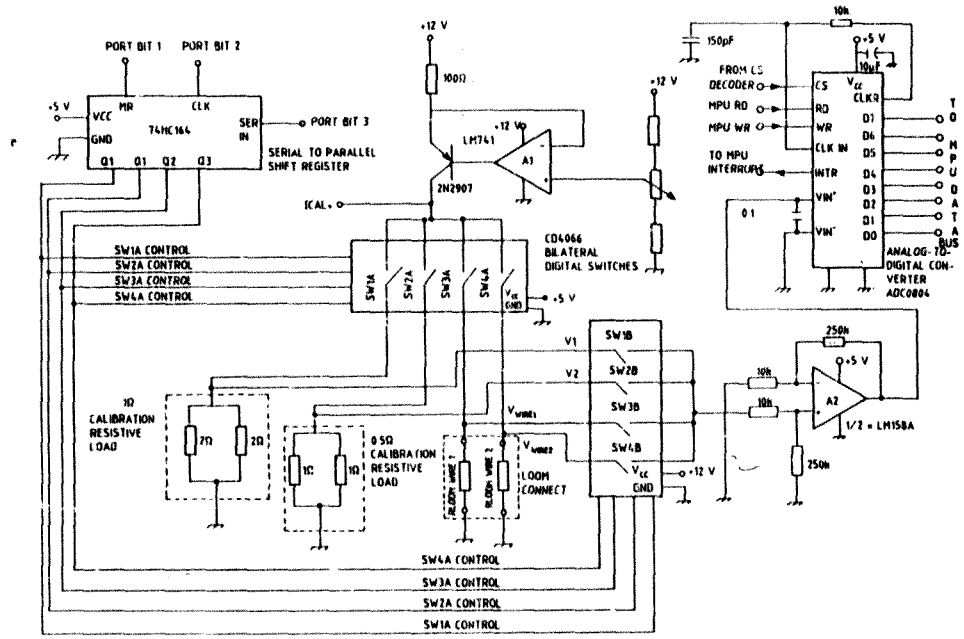


Figure 3.11 Detailed quality measurement card displaying a two-loom wire capacity

Breaking the measurement process down into a series of step-by-step activities, requires a detailed examination of the QUALITY card system.

a. Port bit 1 (MASTER RESET) is toggled, high-low-high, resetting all shift register outputs to the zero condition. As a result, the bilateral switch control lines place all switches, SW1A to SW4B, in the high impedance mode. At this stage the 20 mA current source can be calibrated, placing a precision ammeter across I_{CAL}+ and GND, if so desired. This is the only manual calibration requirement of the measurement system, possibly needing attention on a monthly basis.

b. Port bit 3 (SERIAL INPUT) is pulled high.

c. Port bit 2 (CLOCK INPUT) is toggled, low-high-low, shifting the logic 'high' serial input bit to output Q0.

d. The serial input bit is then tied low for the rest of the measurement cycle. With this configuration, by merely pulsing the clock input bit (port bit 2), the logic high bit present at Q0 can be uniquely rippled through to each of the other outputs (Q1 to Q3). By controlling the position of the ripple effect each digital switch can be individually opened and closed, allowing any of resistive loads (loom wires) to be accurately measured. By simply cascading several shift registers together, the ripple effect can be extended to control any number of digital switches. This microprocessor driven switch control technique allows the system capacity and the number of loom wires measured, to be easily expanded.

e. Pulling the SW1A and SW1B control lines high, i.e. Q0 being a logic high, connects the 1 ohm calibration resistive load to the current source as well as to the measurement circuitry. From equation (8) the digital equivalent of the signal processed calibration voltage V_{01} is transmitted to the MPU.

f. Toggling the clock input (port bit 2), shifts the logic high bit to output Q1. Since the serial input is permanently tied low at this stage, only one of the shift register outputs can be high, allowing only one of the digital switches to be closed at any one time. The 0,5 ohms calibration resistive load is now coupled to the current source and measurement circuitry. Similarly the byte value of the signal processed calibration voltage V_{02}, equation (9), is stored to the MPU.

g. With the relevant automatic calibration information known, i.e. V_1, V_2, V_{01} and V_{02}, software manipulation internal to the microprocessor calculates G' and V'_{OFFSET} as shown in equations (10) and (11).

h. Port bit 2 is toggled again, now coupling loom wire 1 into the measurement system. The signal conditioned voltage of the wire $V'_{WIRE\ 1}$ is transformed into an equivalent digital signal and passed to the microprocessor, via its data bus. The microprocessor then calculates the true resistance value $R_{WIRE\ 1}$ according to the following equations, to within an accuracy of approximately 40 milli-ohms.

$$V'_{WIRE\ 1} = G'V_{WIRE\ 1} + V'_{OFFSET} \qquad \text{equation (14)}$$

$$\therefore V_{WIRE\ 1} = \left[\frac{V'_{WIRE\ 1} - V'_{OFFSET}}{G'}\right] \qquad \text{equation (15)}$$

$$\text{giving } R_{WIRE\ 1} = V_{WIRE\ 1}/I_{CURRENT\ SOURCE} \qquad \text{equation (16)}$$

Via an RS232-C serial port the measurement result can be transmitted to a printer or a terminal.

i. Toggle port bit 2 switching loom wire 2 into the measurement system. The wire resistance value $R_{WIRE\ 2}$ is then determined in the same way as that for loom wire 1.

This completes the detailed measurement cycle of a two-wire QUALITY measurement card.

4 THE CENTRAL PROCESSING UNIT

A Central Processing Unit must be provided to communicate with the measurement cards, namely the QUALITY and GO/NO-GO cards. This implies that control stimuli must be transmitted and that measurement and status data must be received, from these application cards. A CPU is also required to allow the user to interact with the automatic loom tester, by means of a front panel.

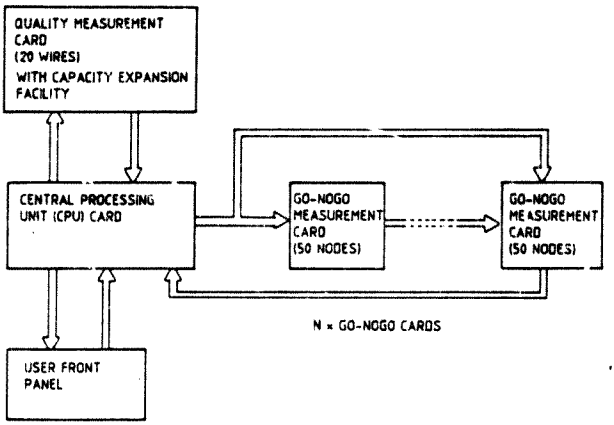


Figure 4.1 The overall system hardware structure

4.1 Choice of Microcomputer and Software Development Tools

The 8031 single component eight-bit microcomputer was chosen as the system controller. This chip provides a cost-effective (approximately R10) solution for those controller applications requiring up to 64 kilo-bytes of program and/or 64 kilo-bytes of data storage. Specifically, the 8031 contains 128 bytes of read/write data memory; 32 I/O lines configured as four eight-bit parallel ports; two 16-bit timer/counters; a five-source, two-priority level nested interrupt structure; a programmable serial I/O port; and an on-chip oscillator with clock circuitry.

The 8031 microcomputer is efficient in both control and computational type applications. This results from extensive BCD/binary arithmetic and bit handling facilities. The controller makes efficient use of its program memory space with an instruction set consisting of 44% one-byte, 41% two-byte and 15% three-byte instructions. When the CPU is operating at 12 MHz, over half of the instructions execute in just 1 micro-second, while the longest instructions, namely multiplication and division, require only 4 micro-seconds. This type of efficiency enhances speed in both the acquisition and the processing of measurement data; an important constraint when testing large looms.

Since the 8031 microcomputer chip is a member of the MCS 51 family, the PL/M-51 language was chosen as the software development tool. High-level languages are more closely modelled on human thought processes than lower level languages, such as Assembler. High-level languages require one less translation step from concept-to-code than do lower-level languages. Consequently, high-level languages are relatively easy to write and can be written faster than low-level languages. Programs in high-level languages are easier to read and understand than those in lower-level languages, and are thus easier to modify. High level languages therefore result in lower costs for both the development and maintenance of programs.

4.2 The User Front Panel

The front panel basically consists of momentary pushbutton switches, Light Emitting Diode (LED) indicators, an RS232-C serial port connector and universal EPROM holder. Figure 4.2 is a detailed representation of the front panel.

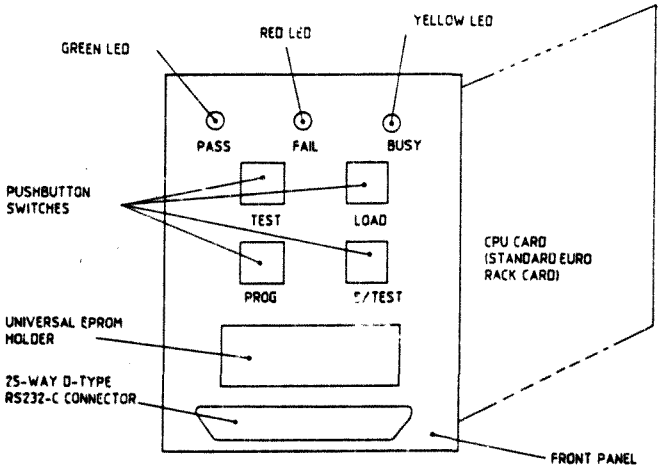


Figure 4.2 Front panel

The BUSY (yellow) LED indicates that a particular operation is being computed. This effectively disconnects the front panel from the user. The PASS (green) LED indicates that a particular operation passed without fault. The FAIL (red) LED indicates that the particular operation has failed. A printer is connected to the serial port (RS232-C) connector, providing a hardcopy of the test results.

4.2.1 The S/TEST pushbutton

The S/TEST pushbutton provides a facility for the GO/NO-GO measurement card/s to be self tested. This ensures that there are no machine-generated short or open circuit fault conditions, thus preventing the constant corruption of measurement data. The self test consists of two phases, a short-circuit check followed by an open-circuit test, already explained in some detail. The self test procedure is as follows:

a. Press the S/TEST pushbutton.

b. The BUSY (yellow) LED will illuminate.

c. The go/no-go card is checked for short-circuit faulty conditions. If failure occurs the faulty node line numbers are printed out. This completes the short-circuit test.

d. At this stage the BUSY LED remains illuminated (flashing) indicating that the S/TEST operation is still in process, i.e. that the open circuit self test must still be performed.

e. The open-circuit test is initiated by first connecting the 'self test loom' to the GO/NO-GO card interface, then pressing the S/TEST pushbutton once again.

f. The go/no-go card is checked for open-circuit conditions. If it fails, the faulty node line numbers will print out, thus completing the open-circuit self test.

4.2.1 The S/TEST pushbutton

The S/TEST pushbutton provides a facility for the GO/NO-GO measurement card/s to be self tested. This ensures that there are no machine-generated short or open circuit fault conditions, thus preventing the constant corruption of measurement data. The self test consists of two phases, a short-circuit check followed by an open-circuit test, already explained in some detail. The self test procedure is as follows:

a. Press the S/TEST pushbutton.

b. The BUSY (yellow) LED will illuminate.

c. The go/no-go card is checked for short-circuit faulty conditions. If failure occurs the faulty node line numbers are printed out. This completes the short-circuit test.

d. At this stage the BUSY LED remains illuminated (flashing) indicating that the S/TEST operation is still in process, i.e. that the open circuit self test must still be performed.

e. The open-circuit test is initiated by first connecting the 'self test loom' to the GO/NO-GO card interface, then pressing the S/TEST pushbutton once again.

f. The go/no-go card is checked for open-circuit conditions. If it fails, the faulty node line numbers will print out, thus completing the open-circuit self test.

g. The BUSY LED switches off, and depending on whether either of the self tests failed or both passed, the FAIL (red) or PASS (green) LED will illuminate, respectively.

4.2.2 The LOAD pushbutton

The LOAD pushbutton provides a hardcopy output of the entire wiring configuration of any loom connected to the tester. This facility has two main advantages:

a. In research and development environments, single one-off looms are commonly found. The verification of such looms will require a printout of the entire wiring pattern, displaying all short-circuit paths by means of node (connector pin) numbers. This information can then visually be compared with the original wiring documentation, allowing the detection of any fault conditions that may be present.

b. In situations where more than one loom of the same type requires testing, verification is effectively achieved by automatically comparing wiring configurations transparant to the user. This involves permanently storing a perfect wiring pattern, of the particular loom type, in an EPROM. The wiring configuration of any future loom of this type will then be compared against the EPROM data, with differences implying fault conditions. If a model loom, i.e. the reference wiring pattern to be permanently stored, is hand tested, there will always be some degree of uncertainty, especially in the more complex configurations.

By continuously printing out the wiring pattern and eliminating faults, the perfect configuration will eventually be obtained, and no uncertainty will exist.

The load procedure is as follows:

a. Connect the desired loom to the machine interface.

b. Press the *LOAD* pushbutton.

c. The *BUSY LED* will illuminate.

d. The wiring configuration of the connected loom is determined and printed out. Only the short-circuit paths are displayed by means of node (connector pin) numbers.

e. The BUSY LED switches off, once again allowing the user access to the front panel.

4.2.3 The PROG pushbutton

Once the user is satisfied with a model reference wiring configuration, the information can be permanently stored in an EPROM. Before inserting a valid data marker, a storage check is performed. If information is found to be stored incorrectly, the EPROM storage space indexer moves to the next blank sector. By pressing the PROG pushbutton again the same reference pattern information is stored and checked in the new sector. This procedure can be continued until either a valid data storage check is obtained or the EPROM storage space indexer exceeds the capacity of the EPROM.

The program procedure is as follows:

a. Insert a blank EPROM into the universal holder located on the front panel.

b. Press the PROG pushbutton. Once the model reference wiring pattern has been obtained, via the load mode, the model loom need not be connected to the tester during the programming procedure.

c. The BUSY LED will illuminate.

d. The reference wiring pattern information is stored in the EPROM and verified.

e. The BUSY LED switches off.

f. If a valid check is obtained the PASS (green) LED will illuminate, or conversely if it fails, the FAIL (red) LED will illuminate.

4.2.4 The TEST pushbutton

This facility determines the current wiring configuration of the loom under test, comparing it to a reference wiring pattern of the same type, stored in EPROM. Any differences found are analysed by the short and open-circuit algorithms, after which the fault diagnosis is displayed in the form of a printout.

The test procedure is as follows:

a. Insert the relevant EPROM, pertaining to the particular loom type under test, into the universal holder located on the front panel.

b. Press the TEST pushbutton.

c. The BUSY (yellow) LED will illuminate.

d. The model EPROM based reference pattern is loaded into the machine.

e. The wiring configuration of the UUT is determined.

f. The UUT wiring pattern is compared to the reference pattern, detecting all short-circuit fault conditions.

g. Fault conditions are printed out in the form of node (connector pin) numbers.

8 The UUT wiring pattern is compared against the reference pattern, detecting all open-circuit fault conditions.

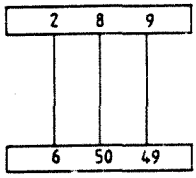
9 Fault conditions are printed out in the form of node (connector pin) numbers.

10 The BUSY led switches off.

11 If any fault conditions existed the FAIL (red) led is illuminated, else the PASS (green) led illuminates.

4.2.5 Hard copy example

The following loom test printout is based on an example shown in Figure 4.3. Part a. exhibits the model reference wiring configuration of type loom A stored on eprom. Part b. shows a faulty wiring pattern of loom A, to be test ' and diagnosed.



a. Loom A

SHORTS

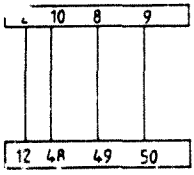
10 TO 48
09 TO 50
08 TO 49
02 TO 12

04 FAIL

OPENS

02 TO 06
08 TO 50
09 TO 49

03 FAIL



b. Faulty loom A

Figure 4.3 Hard copy example

By comparing the two wiring patterns, the seven faults displayed in the hard copy can be quickly verified. As shown above, a major advantage of the fault-finding algorithms of this tester is that multiple fault conditions are fully diagnosed on the first run.

4.3 CENTRAL PROCESSING UNIT HARDWARE

In order to meet the requirements of the S/TEST, TEST, PROG and LOAD modes, the CPU card must provide the following hardware blocks:

a. A microcomputer controller for manipulating and diagnosing measurement results, transmitting/receiving control stimuli/status information to and from the application measurement cards, as well as the overall control of the central processing unit.

b. A monitor Read Only Memory (ROM) containing the entire operating system, ultimately providing the functionality of the tester.

c. An I/O port latch expander facility allowing the controller access to the front panel. This gives the microprocessor more time to handle the more important tasks.

d. External Random Access Memory (RAM) necessary to temporarily store loom wiring patterns, as well as other intermediate variables.

e. An EPROM burner for the permanent storage of model reference wiring configurations.

f. Due to the hardcopy result output, an RS232-C driver/receiver is necessary.

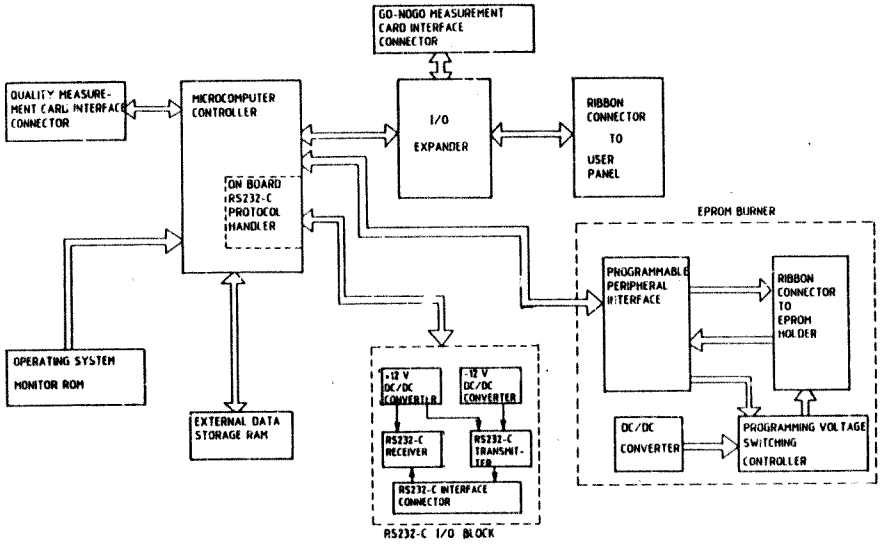


Figure 4.4 Central Processing Unit (CPU) card hardware architecture

Refer to Appendix B for the detailed CPU card hardware schematic. The self explanatory nature of the CPU card hardware leaves only the EPROM burner for discussion. Hardware structure techniques similar to that displayed in the CPU card, are extensively discussed in many advanced microprocessor design textbooks and handbooks.

4.3.1 The EPROM burner

The EPROM burner consists of four basic hardware blocks:

1 The EPROM.
2 A programmable peripheral interface.
3 A DC/DC converter.
4 A programming voltage switching controller.

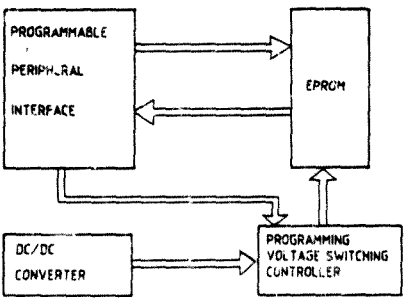


Figure 4.5 EPROM burner hardware block diagram

The EPROM modes of concern are the READ and PROGRAMMING cycles. Access to these operations is achieved by controlling the relevant lines incorporated in the EPROM control bus. Mode of operation selection as well as supplying and receiving of stable address and data information to and from the EPROM, embody the function performed by the Programmable Peripheral Interface (PPI). In the READ mode the programming power supply sits at +5 V, whereas in the PROGRAMMING mode it must sit at +25 V. The programming voltage switching controller ensures this operation. In order to achieve the necessary 25 V programming voltage from a general 5 V system power supply a DC/DC (5 V/25 V) converter step-up configuration is required. The following analysis refers to the detailed schematic of the EPROM burner, shown in Figure 4.6.

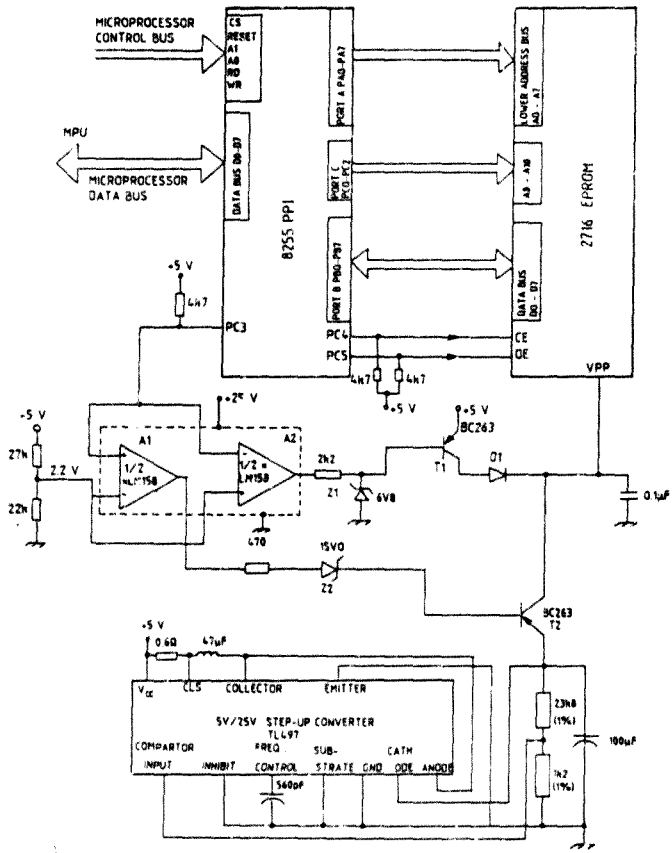


Figure 4.6 Detailed EPROM burner schematic

During power-on the PPI reset input goes 'high', due to the action of the MPU reset configuration. As a result ports A, B and C will be configured in their input modes, (i.e. all port lines will be in the high impedance state. With pull-up resistors on specified port lines, the following initialization will result during power-on:

a. The chip enable $\overline{CE}$ (device selection) is pulled 'high', placing the EPROM in a high impedance state. This effectively disconnects the chip from the rest of the system.

b. Port line PC3, the programming voltage switching controller control line, is pulled high causing the programming voltage supply at Vpp to sit at 5 V. This prevents erroneous data from being burnt into the EPROM every time the tester is switched on.

At this point it must be decided as to whether information is to be read (verified) from or written into the EPROM.

4.3.1.1 A write cycle

Assuming the data byte 77H is to be permanently stored into the EPROM at address location 0001 H, the following procedure will take place:

a. Place ports A, B and C in their output modes. It should be noted that any of the eight bits of ports C only can be set or reset using a single output instruction. This feature reduces software requirements in control-based applications.

b. Write 77H into port B, effectively driving the required data onto the EPROM data bus.

c. Write 01H into port A and reset port lines PC0(A8), PC1(A9) and PC2(A10). This places the required address onto the EPROM address bus, A10A9A8A7A6A5A4A3A2A1A0 = 00000000001B.

d. Reset the PC3 port bit line, switching Vpp to 25 V. The EPROM programming mode requires the Vpp power supply to sit at 25 V and the output enable line $\overline{OE}$ pulled 'high'. Keeping $\overline{OE}$ high gates data from the output pins (D0-D7) preventing bus contention between the EPROM data bus and port B on the PPI. Although damage may result, such a condition will corrupt the programming data.

e. Once the information on the EPROM address and data buses are stable and Vpp = 25 V, the data byte 77H is ready to be burnt into the EPROM.

f. By merely placing a 50 ms active high pulse on the chip enable line $\overline{CE}$, 77H is permanently stored at address 0001H.

4.3.1.2 A read cycle

Assuming a data byte burnt into a specific EPROM address location is to be verified (read), the following procedure is executed:

a. Configure ports A and C in their output modes.

b. Place the specified EPROM address to ports A and C, i.e. A0 → A10.

c. Set port bit PC3 resulting in Vpp = 5 V.

d. Configure port B in its input mode. This allows the MPU, via the PPI, to read the EPROM data.

e. Reset port bit PC5, the output enable control line $\overline{OE}$, g ting data to the EPROM output pins (D0-D7).

f. Instituting a port B read cycle, data corresponding to the specified EPROM address is written into the microprocessor and processed.

4.3.1.3 The programming voltage switching controller operation

As already defined the function of this controller is to switch the programming voltage line Vpp between 5 V and 25 V depending on the required mode of operation. Setting port bit PC3 (approximately 5 V) has the following effect on the controller:

a. Comparator output A1 sits at approximately 25 V.

b. Comparator output A2 sits at approximately 0 V.

c. The 'high' output of A1 places zener diode Z2 in its high impedance blocking region. Being a current sensitive device transistor T2 is cut-off. Hence the 25 V DC/DC converter supply, via transistor T2, becomes disconnected from pin Vpp.

d. The low output of A2 forward biases transistor T1s emitter base junction, switching the 5 V supply to pin Vpp.

Similarly by resetting port bit PC3 (approximately 0 V):

a. Comparator output A1 sits at approximately 0 V.

b. Comparator output A2 sits at approximately 25 V.

c. A 'high' output on A2 causes zener diode Z1 to avalanche. The 6,8 V reference reverse biases transistor T1, resulting in cut-off. A 6V8 zener diode was chosen to achieve a reverse bias condition without stressing the base-emitter junction, otherwise possibly damaging the transistor.

d. The 'low' output of comparator A1 results in the breakdown of zener diode Z2. This condition forward biases transistor T2, connecting the 25 V supply to pin Vpp.

The exclusion of diode D1 would prevent the controller from switching the 25 V supply to pin Vpp, when PC3 is reset. This condition is due to base collector junction breakdown in transistor T1. With diode D1 removed and PC3 reset, results in Vpp ~ 7,5 V and not the expected 25 V. Representing transistor T1 as a diode model Figure 4.7 clearly illustrates this fault condition. Due to the 6,8 V base reference voltage, the base-collector junction of transistor T1 becomes forward biased causing Vpp to sit at approximately 7,5 V. Transistor T2 moves out of saturation in order to absorb the excess potential.

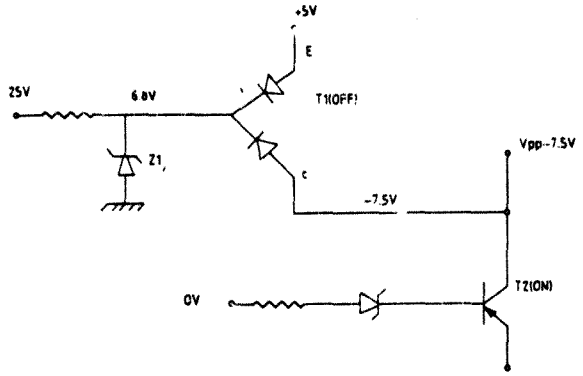


Figure 4.7 Programming voltage effect with switching controller control line PC3 reset and diode D1 removed

Inserting diode D1 in transistor T1s collector path prevents the forward bias effect of its base-collector junction, thus ensuring the required 25 V programming voltage. The functioning and design equations of the DC/DC step-up converter can be obtained from the Texas Instruments Power data manual or the equivalent thereof.

4.3.2 The operating system

Although the 8031 micro-controller constitutes the nucleus of entire testing system, the functionality and intelligence of the loom tester is provided by the firmware stored in the monitor EPROM. The following sequence of flow diagrams fully describe the operating system with finer detail found in the software listings contained in Appendix C.

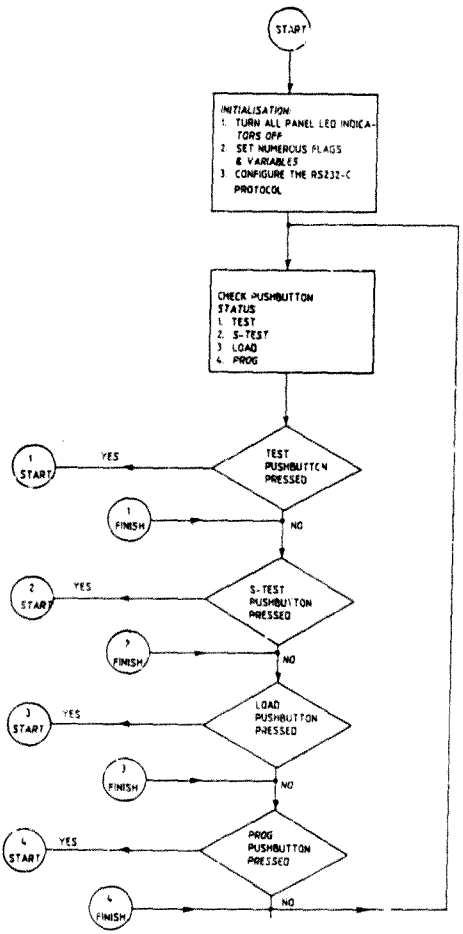


Figure 4.8 Main program loop flow diagram

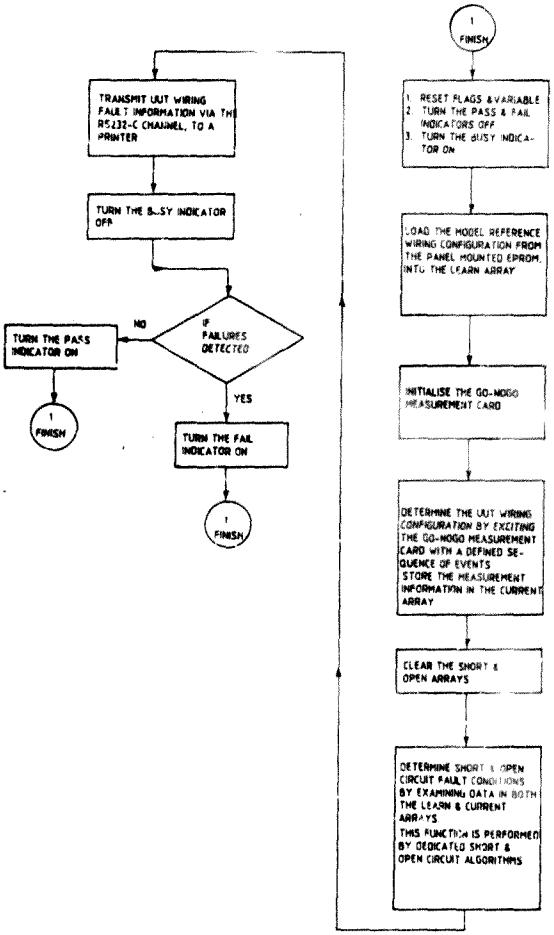


Figure 4.9 TEST mode flow diagram

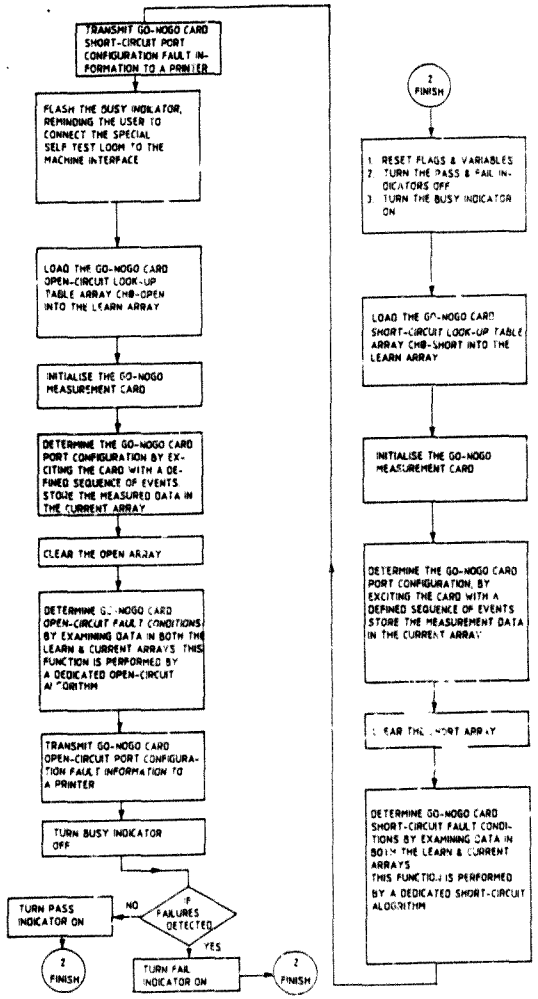


Figure 4.10 S-TEST mode flow diagram

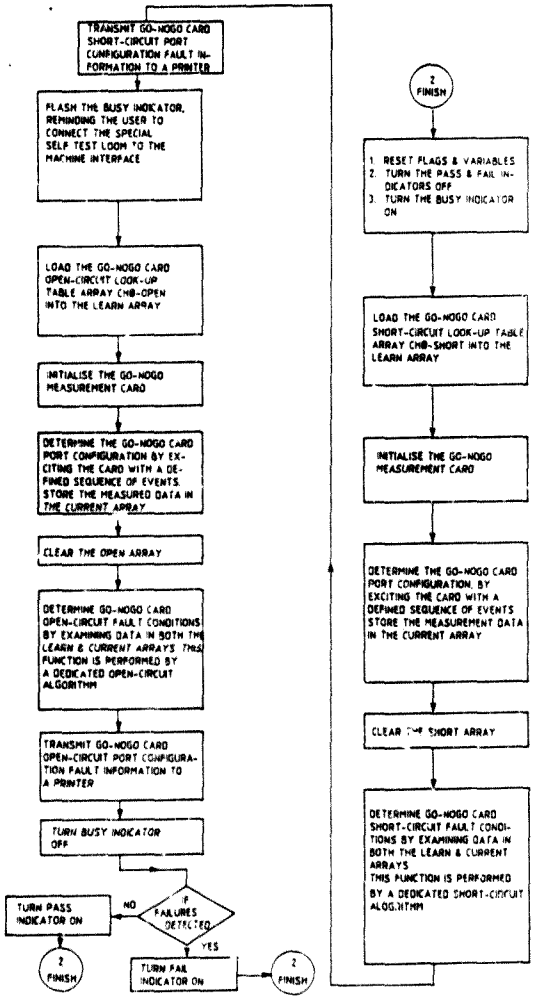


Figure 4.10 S-TEST mode flow diagram

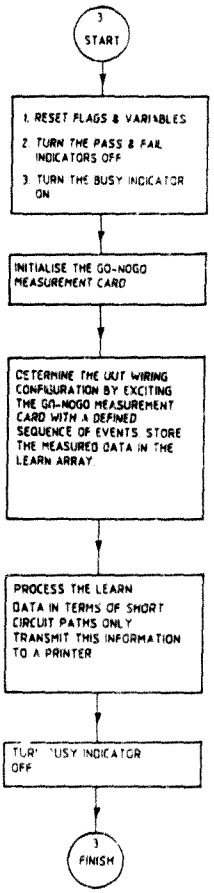


Figure 4.11 LOAD mode flow diagram

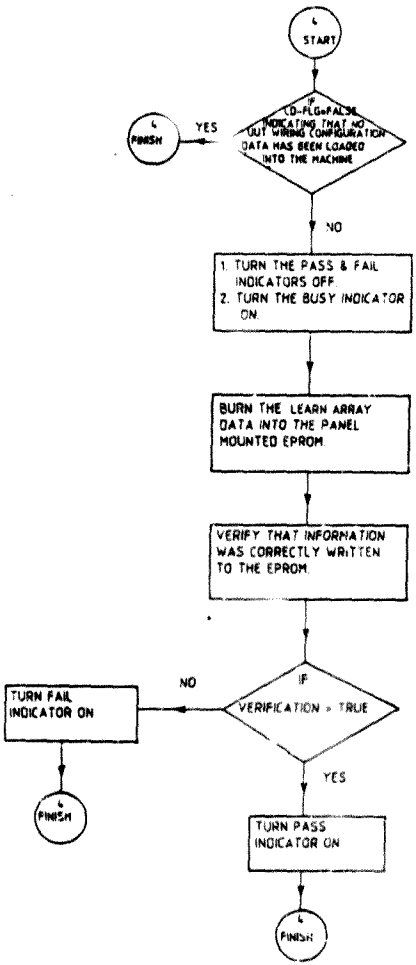


Figure 4.1L PROG mode flow diagram

5 CONCLUSION

Extensive work has been done investigating and developing loom testing requirements. This has resulted in a low-cost microprocessor based loom testing system. Appendix D illustrates the current prototype tester.

Beside being able to play a vital role in industry today, the AUTOMATIC LOOM TESTER discussed the only known one of its kind to be developed in South Africa. Although there has been much discussion pertaining to the basic design of such a machine, areas encompassing future ideas have been left untouched. In conclusion, probable future work on the loom tester will be discussed.

At present the prototype rack-based loom tester allows go/no-go type testing only, having a fully industrialised go/no-go measurement card. Although the design concept concerning POWER-LINE type measurement techniques has been proven, the QUALITY measurement card has not yet been integrated into the overall testing system. This leaves areas such as:

a. the machine interface;
b. resistance errors due to UUT/machine adaptor harnesses,

still to be investigated. Even though a four-wire resistance measurement is effectively being performed on each loom wire, the added error introduced by an adaptor harness is illustrated in Figure 5.1.

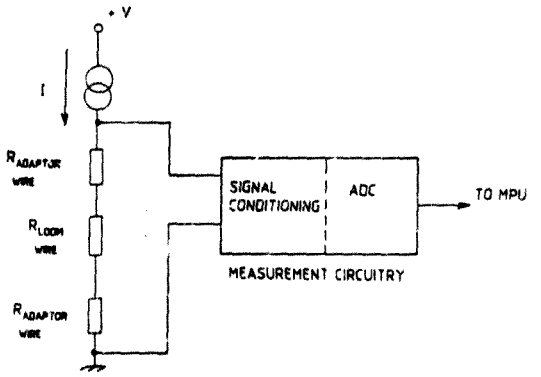


Figure 5.1 Lumped resistance error due to adaptor harness

Due to the necessity for a QUALITY measurement card, ways of accurately compensating for such conditions will be thoroughly investigated.

Situations often arise where looms and cables must be tested 'on site.' One of the original requirements was to provide some sort of portability option. This will call for investigations into the following topics:

a. secondary cells, possibly nickel-cadmium cells;
b. battery chargers;
c. battery regulation cards;
d. power consumption characteristics of the tester;
e. noise generation effects and remedies due to power switching.

Providing a stable, noise-free power supply is fundamentally important to the successful implementation of a microcomputer system. The decision whether to buy or to manufacture a power supply will basically depend on factors such a cost, time and availability. Although a demonstration power supply, 5 V $\pm$ 5% 3 A, has been designed to specifically slide into a standard rack housing, no final decision has been taken concerning the power supply module.

At present, only one model reference wiring configuration can be stored per EPROM. Fault conditions and wiring patterns are printed out in terms of node numbers, which must then be manually converted to the real loom connector notation. Making use of the RS232-C serial port facility a terminal could link into the test system, greatly extending the user/machine interaction. By expanding the operating system firmware in conjunction with a terminal connection, the following possibilities exist:

a. allowance for the user to store many model reference wiring configurations per EPROM by creating a directory facility;

b. allowance for the user to manually edit and store model reference wiring configurations without the need to connect a loom to the tester interface;

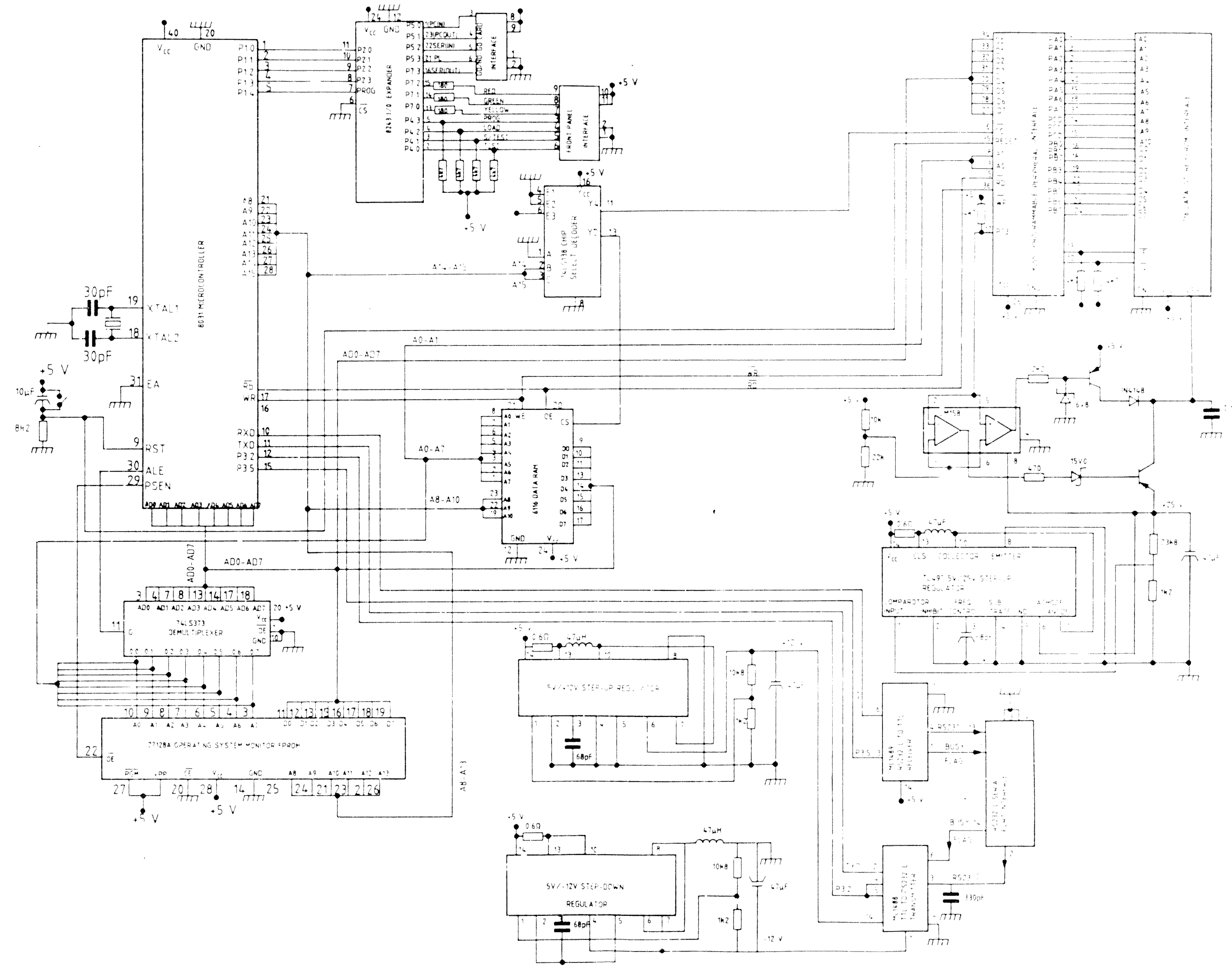
c. the ability to enter and store loom connector names in EPROM resulting in fault conditions and wiring patterns being represented by the actual connector pin notation;

d. allowance for the user to choose and halt any single fault condition or short-circuit combination, exposing intermittent faults.

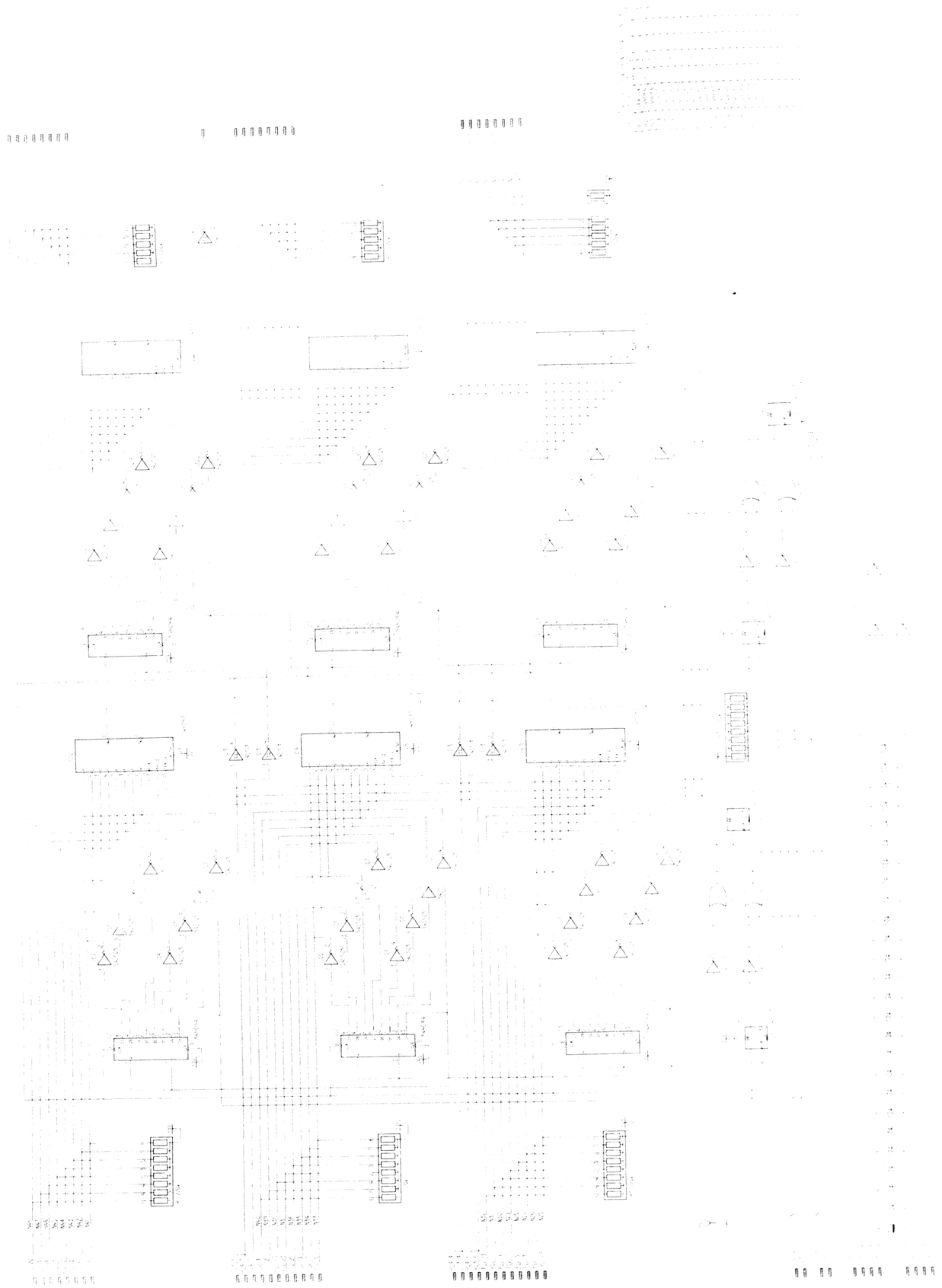
In time, many more ideas and developments will come to light. A seed has been sown, with only the needs of the customer to determine how it grows.

APPENDIX A

CIRCUIT DIAGRAM OF THE GO/NO-GO MEASUREMENT MODULE


8031 MICROCONTROLLER
8243 I/O EXPANDER
INTERFACE
FRONT PANEL INTERFACE
74LS138 CHIP SELECT DECODER
6116 DATA RAM
74LS373 DEMULTIPLEXER
27128A OPERATING SYSTEM MONITOR EPROM
5V/+12V STEP-UP REGULATOR
5V/-12V STEP-DOWN REGULATOR
XTAL1
XTAL2
30pF
+5 V
AD0-AD7
A0-A7
A8-A10
A8-A13
A14-A15

163(a)



APPENDIX B

CIRCUIT DIAGRAM OF THE CENTRAL PROCESSING UNIT

```
START:
DO;
 DECLARE DEC LITERALLY'DECLARE';
  DEC REG LITERALLY'REGISTER';
 DEC FALSE LITERALLY'0';
  DEC TRUE LITERALLY'1',SIZE LITERALLY'50';
 DEC SER_OUT BIT AT(95H) REG;
  DEC P1 BYTE AT(90H) REG;
 DEC P2 BYTE AT(0A0H) REG;
  DEC TCON BYTE AT(88H) REG,TMOD BYTE AT(89H)REG;
 DEC SCON BYTE AT(98H) REG,SBUF BYTE AT(99H) REG;
  DEC TH1 BYTE AT(8DH) REG,INT BYTE AT(0A8H) REG;
 DEC PORT3_5 BIT AT(0B5H) REG;
  DEC CONJ(4) BYTE CONSTANT(' TO ');
 DEC A_NO(2) BYTE;
 DEC(I,I_NO,L_NO,REC_NO,NUMBER,HSD,LEN_ANO,STAR,FINI,DATA_1) BYTE;
  DEC FOREVER LITERALLY'WHILE 1';
 DEC S_DATA(*) BYTE CONSTANT(20H,20H,'SHORTS',0DH,0DH);
  DEC O_DATA(*) BYTE CONSTANT(20H,20H,'OPENS',0DH,0DH);
 DEC P_DATA(*) BYTE CONSTANT(20H,20H,'PASS',0DH,0DH);
  DEC F_DATA(*) BYTE CONSTANT(20H,20H,'FAIL',0DH,0DH);
 DEC(COUNT,NODE$NO,DUMMY,COUNT$2,COUNT$1,PORT_2,PORT_3) BYTE;
  DEC LEARN$DATA(50) BYTE AT(6000H) AUXILIARY;
 DEC(CHECK,MULT,REAL_NODE,LARGEST) BYTE;
  DEC(MARK_1,MARK_2,COUNT$3,COUNT$4,BIG_NUM,COUNT$5,COUNT$6) BYTE;
 DEC LEARN(SIZE) BYTE AT(6000H+32H) AUXILIARY;
  DEC CURRENT(SIZE) BYTE AT(6000H+32H+SIZE) AUXILIARY;
 DEC(RED,CUR_FLAG,CHECK_1) BYTE;
  DEC OPEN(SIZE) BYTE AT(6000H+32H+SIZE+SIZE) AUXILIARY;
```

APPENDIX C

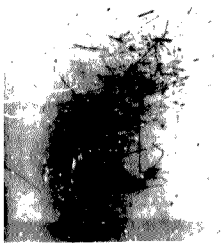
SOURCE LISTING OF THE OPERATING SYSTEM

```
START;
DO;
 DECLARE DEC LITERALLY'DECLARE';
  DEC REG LITERALLY'REGISTER';
 DEC FALSE LITERALLY'0';
  DEC TRUE LITERALLY'1',SIZE LITERALLY'50';
 DEC SER_OUT BIT AT(95H) REG;
  DEC P1 BYTE AT(90H) REG;
 DEC P2 BYTE AT(0A0H) REG;
  DEC TCON BYTE AT(88H) REG,TMOD BYTE AT(89H)REG;
 DEC SCON BYTE AT(98H) REG,SBUF BYTE AT(99H) REG;
  DEC TH1 BYTE AT(8DH) REG,INT BYTE AT(0A8H) REG;
 DEC PORT3_5 BIT AT(0B5H) REG;
  DEC CONJ(4) BYTE CONSTANT(' TO ');
 DEC A_NO(2) BYTE;
 DEC(IL_NO,L_NO,REC_NO,NUMSEK,HSD,LEN_AND,STAR,FINI,DATA_1) BYTE;
  DEC FOREVER LITERALLY'WHILE 1';
 DEC S_DATA(*) BYTE CONSTANT(20H,20H,'SHORTS',0DH,0DH);
  DEC O_DATA(*) BYTE CONSTANT(20H,20H,'OPENS',0DH,0DH);
 DEC P_DATA(*) BYTE CONSTANT(20H,20H,'PASS',0DH,0DH);
  DEC F_DATA(*) BYTE CONSTANT(20H,20H,'FAIL',0DH,0DH);
 DEC(COUNT,NODE$NO,DUMMY,COUNT$2,COUNT$1,PORT_2,PORT_6) BYTE;
  DEC LEARN$DATA(50) BYTE AT(6000H) AUXILIARY;
 DEC(CHECK,MULT,REAL_NODE,LARGEST) BYTE;
  DEC(MARK_1,MARK_2,COUNT$3,COUNT$4,BIG_NUM,COUNT$5,COUNT$6) BYTE;
 DEC LEARN(SIZE) BYTE AT(6000H+32H) AUXILIARY;
  DEC CURRENT(SIZE) BYTE AT(6000H+32H+SIZE) AUXILIARY;
 DEC(RED,CUR_FLAG,CHECK_1) BYTE;
  DEC OPEN(SIZE) BYTE AT(6000H+32H+SIZE+SIZE) AUXILIARY;
```


```
START:
DO;
 DECLARE DEC LITERALLY'DECLARE';
  DEC REG LITERALLY'REGISTER';
 DEC FALSE LITERALLY'0';
  DEC TRUE LITERALLY'1',SIZE LITERALLY'50';
 DEC SER_OUT BIT AT(95H) REG;
  DEC P1 BYTE AT(90H) REG;
 DEC P2 BYTE AT(0A0H) REG;
  DEC TCON BYTE AT(88H) REG,TMOD BYTE AT(89H)REG;
 DEC SCON BYTE AT(98H) REG,SBUF BYTE AT(99H) REG;
  DEC TH1 BYTE AT(8DH) REG,INT BYTE AT(0A8H) REG;
 DEC PORT3_5 BIT AT(0B5H) REG;
  DEC CONJ(4) BYTE CONSTANT(' TO ');
 DEC A_NO(2) BYTE;
 DEC(I,I_NO,L_NO,REC_NO,NUMBER,HSD,LEN_ANO,STAR,FINI,DATA_1) BYTE;
  DEC FOREVER LITERALLY'WHILE 1';
 DEC S_DATA(*) BYTE CONSTANT(20H,20H,'SHORTS',0DH,0DH);
  DEC O_DATA(*) BYTE CONSTANT(20H,20H,'OPENS',0DH,0DH);
 DEC P_DATA(*) BYTE CONSTANT(20H,20H,'PASS',0DH,0DH);
  DEC F_DATA(*) BYTE CONSTANT(20H,20H,'FAIL',0DH,0DH);
 DEC(COUNT,NODE$NO,DUMMY,COUNT$2,COUNT$1,PORT_2,PORT_6) BYTE;
  DEC LEARN$DATA(50) BYTE AT(6000H) AUXILIARY;
 DEC(CHECK,MULT,REAL_NODE,LARGEST) BYTE;
  DEC(MARK_1,MARK_2,COUNT$3,COUNT$4,BIG_NUM,COUNT$5,COUNT$6) BYTE;
 DEC LEARN(SIZE) BYTE AT(6000H+32H) AUXILIARY;
  DEC CURRENT(SIZE) BYTE AT(6000H+32H+SIZE) AUXILIARY;
 DEC(RED,CUR_FLAG,CHECK_1) BYTE;
  DEC OPEN(SIZE) BYTE AT(6000H+32H+SIZE+SIZE) AUXILIARY;
```

```
DEC SHORT(SIZE) BYTE AT(6000H+32H+SIZE+SIZE+SIZE) AUXILIARY;
 DEC(I,L,K,J) BYTE;
DEC(CURRENT$START,CURRENT$STOP,LEARN$START,LEARN$STOP) BYTE;
 DEC(DATA,LOAD,TEST,PROG,S_TEST,LD_FLG) BYTE;
DEC(GAP,COUNT$7,L_ADD,COUNT$8) WORD;
 DEC INFO(4) BYTE AT(0A000H) AUXILIARY;
DEC(H_ADD,READ,N_VERIFY) BYTE;
 DEC CH0_SHORT(*) BYTE CONSTANT(1,2,3,4,5,6,7,8,9,10,11,12,13,14,
15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,
 36,37,38,39,40,41,42,43,44,45,46,47,48,49,50);
DEC C40_OPEN(*) BYTE CONSTANT(26,1,27,2,28,3,29,4,30,5,31,6,32,7,
 33,8,34,9,35,10,36,11,37,12,38,13,39,14,40,15,41,16,42,17,43,18,
44,19,45,20,46,21,47,22,48,23,49,24,50,25);

 /* SER_INT SERVICES THE RS 232-C TRANSMIT INTERRUPT FLAG.THIS FLAG IS
    SET BY HARDWARE AT THE BEGINNING OF THE STOP BIT IN ANY SERIAL TR-
     AMSMISSION.MUST BE CLEARED BY SOFTWARE*/
SER_INT:
PROCEDURE INTERRUPT 4 USING 1;
  DUMMY=0;
  SCON=SCON AND 0FDH;
 END SER_INT;

/* DEL PROVIDES THE NECESSARY DELAY TO PREVENT THE CORRUPTION OF SERIAL
    DATA BITS BEING TRANSMITTED*/
 DEL:
  PROCEDURE;
  DUMMY=1;
   DO WHILE DUMMY=1;
    J=J+1;
```

```
   END;
 END DEL;

/* DEL_1 POLLS THE SERIAL PORT DESTINATION DEVICE FOR BUSY STATUS*/
  DEL_1:
 PROCEDURE;
   DO WHILE PORT3_5=1;
    J=J+1;
   END;
 END DEL_1;

/* SPACE EXECUTES TWO CONSECUTIVE CR+LF COMMANDS*/
  SPACE:
 PROCEDURE;
   DO K=1 TO 2;
    SBUF=20H;
     CALL DEL;
    CALL DEL_1;
   END;
 END SPACE;

/* PASS ENABLES THE WORD 'PASS' TO BE DUMPED TO A PRINTER*/
  PASS:
 PROCEDURE;
   DO I=0 TO LAST(P_DATA);
    SBUF=P_DATA(I);
     CALL DEL;
    CALL DEL_1;
   END;
 END PASS;
```



```
/* FAULT ENABLES THE WORD 'FAULT' TO BE DUMPED TO A PRINTER*/
 FAULT:
 PROCEDURE:
   DO I=0 TO LAST(F_DATA):
    SBUF=F_DATA(I):
     CALL DEL:
    CALL DEL_1:
    END:
 END FAULT:

/* ASC_NO CONVERTS NODE NUMBERS IN HEX TO ASCII NOTATION*/
  ASC_NO:
 PROCEDURE:
   H_NO=SHR((HEC_NO AND 0F0H),4):
    L_NO=HEC_NO AND 0FH:
   NUMBER=(H_NO*10H)+L_NO:
    A_NO(0)=(NUMBER/0AH) OR 30H:
   A_NO(1)=(NUMBER MOD 0AH) OR 30H:
     END ASC_NO:

/* MESSAG_OUT ENABLES TEST RESULT FAULT CONDITION NODE NUMBERS TO BE
     TRANSMITTED IN ASCII NOTATION DOWN SERIAL LINK IN THE FORMAT
                (NODE NO.) TO (NODE NO.)
                 (NODE NO.) TO (NODE NO.)
                       ETC
    MESS_LOC REPRESENTS THE ARRAY CONTAINING THE RELEVANT DATA
     MESS_LEN REPRESENTS THE ARRAY DATA LENGTH
    THE ARRAY OF CONCERN IS SPECIFIED IN THE CALL COMMAND*/
     MESSAG_OUT:
```

```
PROCEDURE(MESS_LOC,MESS_LEN);
  DEC MESS_LOC WORD,MESS_LEN BYTE;
 DEC(MESSAGE BASED MESS_LOC) (1) BYTE AUXILIARY;

  I=0;
 DO WHILE (I<=MESS_LEN) AND (MESSAGE(I)<>0);
   CALL SPACE;
  HEC_NO=MESSAGE(I);
   CALL ASC_NO;
  DO K=0 TO 1;
    SBUF=A_NO(K);
   CALL DEL;
    CALL DEL_1;
  END;
   DO L=0 TO 3;
   SBUF=CONJ(L);
    CALL DEL;
   CALL DEL_1;
   END;
  I=I+1;
   HEC_NO=MESSAGE(I);
  CALL ASC_NO;
   DO K=0 TO 1;
   SBUF=A_NO(K);
    CALL DEL;
   CALL DEL_1;
   END;
  I=I+1;
   SBUF=0DH;
  CALL DEL;
```

```
      CALL DEL_1;
     END;/*DO WHILE*/
       SBUF=0DH;
      CALL DEL;
       CALL DEL_1;
    END MESSAG_OUT;

/* YELLOW_ON ILLUMINATES THE BUSY LED*/
 YELLOW_ON:
  PROCEDURE;
   P1=17H;
    P1=P1 AND 0EFH;
   PORT_2=PORT_2 AND 0EH;
    P1=PORT_2;
   P1=P1 OR 10;
  END YELLOW_ON;

/* RED_ON ILLUMINATES THE FAIL LED*/
  RED_ON:
 PROCEDURE;
    P1=17H;
   P1=P1 AND 0EFH;
    PORT_2=PORT_2 AND 0DH;
   P1=PORT_2;
    P1=P1 OR 10H;
 END RED_ON;

/* GREEN_ON ILLUMINATES THE PASS LED*/
  GREEN_ON:
 PROCEDURE;
```

```
    P1=17H;
    P1=P1 AND 0FFH;
    PORT_2=PORT_2 AND 0DH;
    P1=PORT_2;
    P1=P1 OR 10H;
  END GREEN_ON;

  /*YELLOW_OFF TURNS THE BUSY LED OFF*/
  YELLOW_OFF:
   PROCEDURE;
    P1=17H;
    P1=P1 AND 0EFH;
    PORT_2=PORT_2 OR 01H;
    P1=PORT_2;
    P1=P1 OR 10H;
   END YELLOW_OFF;

 /* RED_OFF TURNS THE FAIL LED OFF*/
   RED_OFF:
  PROCEDURE;
     P1=17H;
    P1=P1 AND 0FFH;
     PORT_2=PORT_2 OR 04H;
    P1=PORT_2;
     P1=P1 OR 10H;
  END RED_OFF;

 /* GREEN_OFF TURNS THE PASS LED OFF*/
   GREEN_OFF:
  PROCEDURE;
```

```
   P1=17H;
  P1=P1 AND 0EFH;
   PORT_2=PORT_2 OR 02H;
  P1=PORT_2;
   P1=P1 OR 10H;
 END GREEN_OFF;

/* TEST_SW SCANS THE TEST SWITCH FOR ANY CLOSURE*/
  TEST_SW:
 PROCEDURE;
   P1=10H;
  P1=P1 AND 0CFH;
   P1=P1 OR 0FH;
  DATA=P1;
   P1=P1 OR 10H;
  TEST=DATA AND 0FH;
  END TEST_SW;

/* S_TEST_SW SCANS THE SELF TEST SWITCH FOR ANY CLOSURE*/
  S_TEST_SW:
 PROCEDURE;
   P1=10H;
  P1=P1 AND 0CFH;
   P1=P1 OR 0FH;
  DATA=P1;
   P1=P1 OR 10H;
  S_TEST=DATA AND 0FH;
  END S_TEST_SW;
/* LOAD_SW SCAN THE LOAD SWITCH FOR ANY CLOSURE*/
```

```
 LOAD_SW:
 PROCEDURE;
  P1=10H;
  P1=P1 AND 0EFH;
  P1=P1 OR 0FH;
  DATA=P1;
  P1=P1 OR 10H;
  LOAD=DATA AND ( H;
 END LOAD_SW;

/* PROG_SW SCAN THE PROGRAM SWITCH FOR ANY CLOSURE*/
 PROG_SW:
 PROCEDURE
  P1=10H;
  P1=P1 AND 0EFH;
  P1=P1 OR 0FH;
  DATA=P1;
  P1=P1 OR 10H;
  PROG_DATA AND 0FH;
 END PROG_SW;

/* ONE_S_DEL PROVIDES A ONE SECOND DELAY*/
 ONE_S_DEL:
 PROCEDURE;
   DO J=1 TO 100;
    CALL TIME(100);
   END;
 END ONE_S_DEL;

/* PROG_EPROM BURNS A MODEL REFERENCE WIRING PATTERN INTO AN EPROM.THE
```

```
   ORIGINAL DATA IS COMPARED WITH THE BURNT DATA.IF VERIFIED A VALID
   DATA STORAGE MARKER '00H' IS BURNT INTO THE EPROM AND AGAIN VERIFIED.
    ANY VERIFICATION FAILURE CAUSES THE EPROM DATA STORAGE MARKER 'GAP'
   TO BE MOVED TO A NEW VACANT DATA STORAGE SECTOR,CONFIGURING THE
    EPROM FOR ANOTHER STORAGE ATTEMPT.THIS PROCEDURE CAN BE REPEATED
   UNTIL EITHER A VALID DATA STORAGE CYCLE IS OBTAINED OR THE STORAGE
    CAPACITY OF THE EPROM IS EXCEEDED*/
PROG_EPROM:
 PROCEDURE;
  COUNT$7=GAP;
   N_VERIFY=TRUE;
  L_ADD=GAP MOD 256;
   H_ADD=GAP/256;
  INFO(3)=30H;
   J=1;
  INFO(2)=28H+H_ADD;
   CALL TIME(100);
  DO I=0 TO (SIZE-1);
    COUNT$7=COUNT$7+1;
   INFO(2)=INFO(2) OR 08H;
    L_ADD=L_ADD+1;
   IF L_ADD=256 THEN
     DO;
      INFO(2)=INFO(2)+1;
       L_ADD=0;
     END;
     INFO(0)=LOW(L_ADD);
    J=1;
     INFO(1)=LEARN(I);
    J=1;
```

```
 INFO(2)=INFO(2) OR 10H;
  DO J=1 TO 50;
   CALL TIME(10);
  END;
 INFO(2)=INFO(2) AND 0EFH;
 END;/*DO I*/
L_ADD=GAP MOD 256;
H_ADD=GAP/256;
N_VERIFY=FALSE;
 INFO(3)=32H;
J=1;
INFO(2)=90H+H_ADD;
 CALL ONE_S_DEL;
DO I=0 TO (SIZE-1);/*READ VERIFY*/
  L_ADD=L_ADD+1;
 IF L_ADD=256 THEN
   DO;
    INFO(2)=INFO(2)+1;
     L_ADD=0;
   END;
  INFO(0)=LOW(L_ADD);
 J=1;
  READ=INFO(1);
 IF READ<>LEARN(I) THEN
   N_VERIFY=TRUE;
 END;/*DO I*/
/*INSERT MARKER 00H AND VERIFY*/
 IF N_VERIFY=FALSE THEN
   DO;
    INFO(3)=80H;
```

```
      L_ADD=GAP MOD 256;
     H_ADD=GAP/256;
      INFO(2)=20H+H_ADD;
     CALL TIME(100);/*SETTLE +25V*/
      INFO(0)=LOW(L_ADD);
     J=1;
      INFO(1)=00H;
     J=1;
     INFO(2)=INFO(2) OR 10H;
      DO J=1 TO 50;
       CALL TIME(10);
      END;
     INFO(2)=INFO(2) AND 0EFH;
      J=1;
     INFO(3)=02H;
      J=1;
     INFO(2)=INFO(2) AND 07'H;
      CALL ONE_S_DEL;
     READ=INFO(1);
      IF READ<>00H THEN
       N_VERIFY=TRUE;
     END;/*IF*/
    GAP=COUNT$7+1;
/*CS=0E=0,VPP=5V,A10=A9=A0=0*/
    INFO(2)=30H;
   END PROG_EPROM;

/* EPROM_LOAD LOCATES THE VALID DATA STORAGE MARKER '00H', THEN LOADS
    THE MODEL REFERENCE WIRING PATTERN DATA INTO THE LEARN ARRAY*/
  EPROM_LOAD:
```

```
  PROCEDURE;
   INFO(3)=02H;
    J=1;
   INFO(2)=00H;
    CALL ONE_S_DEL;
   H_ADD=0;
    L_ADD=0;
   READ=0FFH;
    COUNT$0=0;
   DO WHILE (COUNT$0<=07FFH) AND (READ<>0);
     INFO(2)=H_ADD;
    J=1;
     INFO(0)=LOW(L_ADD);
      J=1;
     READ=INFO(1);
      L_ADD=L_ADD+1;
     IF L_ADD=256 THEN
       DO;
        H_ADD=H_ADD+1;
         L_ADD=0;
       END;
      COUNT$0=COUNT$0+1;
   END; /*DO WHILE*/
/*AT FIRST POSITION OF VALID DATA*/
   DO I=0 TO (SIZE-1);
     INFO(0)=LOW(L_ADD);
    J=1;
     INFO(2)=H_ADD;
    J=1;
     LEARN(I)=INFO(1);
```

```
    L_ADD=L_ADD+1;
     IF L_ADD=256 THEN
     DO;
      H_ADD=H_ADD+1;
       L_ADD=0;
     END;
     END;
    INFO(2)=30H;
   END EPROM_LOAD;


/*CP$IN$CLK TOGGLES THE SERIAL-TO-PARALLEL SHIFT REGISTER CLOCKS*/
  CP$IN$CLK:
 PROCEDURE;
   P1=15H;
  P1=P1 AND 0CFH;
   PORT_G=PORT_G OR 01H;
  P1=PORT_G;
   P1=P1 OR 10H;
  P1=15H;
   P1=P1 AND 0EFH;
  PORT_G=PORT_G AND 0FEH;
   P1=PORT_G;
  P1=P1 OR 10H;
   P1=15H;
  P1=P1 AND 0CFH;
   PORT_G=PORT_G OR 01H;
  P1=PORT_G;
   P1=P1 OR 10H;
 END CP$IN$CLK;
```

```
/* APP_INIT INITIALISES THE GO-NOGO MEASUREMENT CARD, THEN SETS-UP THE
    FIRST TEST-BIT GENERATION NODE AT THE NODE 1 PORT LOCATION*/
 APP_INIT:
  PROCEDURE;
   DO COUNT=1 TO SIZE;
     CALL CP$IN$CLK;
   END;
/*ALL OUTPUTS OF 055 HIGH*/
 /*PULL SER-IN HIGH*/
   P1=15H;
    P1=P1 AND 0CFH;
   PORT_G=PORT_G AND 0FBH;
    P1=PORT_G;
   P1=P1 OR 10H;
    CALL CP$IN$CLK;
/*PULL SER-IN LOW*/
    P1=15H;
   P1=P1 AND 0CFH;
    PORT_G=PORT_G OR 04H;
   P1=PORT_G;
    P1=P1 OR 10H;
 END APP_INIT;

/* GO$NOGO$TEST DETERMINES THE WIRING CONFIGURATION OF THE LOOM
    CURRENTLY INTERFACED TO THE TESTER. DATA IN THE FORM OF NODE NUMBERS
    ARE STORED IN THE 'CURRENT' ARRAY*/
  GO$NOGO$TEST:
  PROCEDURE;
   NODE$NO=1; /*TEST NODE POINTER*/
```

```
 COUNT$4=0; /*LEARN$ARRAY POINTER*/
 IF CUR_FLAG=TRUE THEN
   DO ;
    DO COUNT=0 TO (SIZE-1);
      CURRENT(COUNT)=0;
    END;
    END;
 ELSE
    DO;
    DO COUNT=0 TO (SIZE-1);
      LEARN(COUNT)=0;
    END;
    END;
 DO WHILE (NODE$NO<=SIZE);
 /*DELAY TO COUNTER TRANSMISSION LINE EFFECTS*/
   DO COUNT=1 TO 100;
     DUMMY=DUMMY OR 01H;
   END;
/*LOAD PATTERN INTO OUTPUT REGISTERS VIA PL*/
    P1=15H;
   P1=P1 AND 0EFH;
    PORT_G=PORT_G AND 0F7H;
   P1=PORT_G;
    P1=P1 OR 10H;
   P1=15H;
    P1=P1 AND 0EFH;
   PORT_G=PORT_G OR 08H;
    P1=PORT_G;
   P1=P1 OR 10H;
    P1=15H;
```

```
P1=P1 AND 0EFH;
 PORT_0=PORT_0 AND 07FH;
P1=PORT_0;
 P1=P1 OR 10H;
COUNT$1=1; /*SHORTED NODES POINTER*/
 COUNT$2=0;
MARK_1=COUNT$2;
 DO COUNT=1 TO SIZE;
  CALL TIME(10);
   IF SER_OUT  THEN
   DO;
    CHECK_1=COUNT$1 MOD 50;
     CHECK=COUNT$1 MOD 8;
    MULT=COUNT$1/8;
     IF CHECK=0 THEN
     REAL_NODE=(((MULT*8)+1)-(((MULT+1)*8)-COUNT$1));
     ELSE
     REAL_NODE=(((MULT+1)*8)-COUNT$1)+((MULT*8)+1);
     IF (CHECK_1=49) OR (CHECK_1=0) THEN
     REAL_NODE=COUNT$1;
     IF COUNT$2<=32H THEN
     DO;
      LEARN$DATA(COUNT$2)=REAL_NODE;
       COUNT$2=COUNT$2+1;
     END;
    END;
 /*TOGGLE CP-OUT*/
   P1=15H;
  P1=P1 AND 0EFH;
```

```
PORT_G=PORT_G OR 02H;
P1=PORT_G;
 P1=P1 OR 10H;
P1=15H;
 P1=P1 AND 0EFH;
PORT_G=PORT_G AND 0FDH;
 P1=PORT_G;
P1=P1 OR 10H;
 P1=15H;
P1=P1 AND 0EFH;
 PORT_G=PORT_G OR 0_H;
P1=PORT_G;
 P1=P1 OR 10H;
COUNT$1=COUNT$1+1;
END;
MARK_2=COUNT$2;
IF MARK_1<>MARK_2 THEN
DO;
 DO COUNT$3=MARK_1 TO (MARK_2-1);
   LARGEST=COUNT$3;
  BIG_NUM=LEARN$DATA(COUNT$3);
   DO COUNT=MARK_1 TO (MARK_2-1);
    IF BIG_NUM<LEARN$DATA(COUNT) THEN
      DO;
       BIG_NUM=LEARN$DATA(COUNT);
        LARGEST=COUNT;
      END;
     END;
   IF CUR_FLAG=TRUE THEN
     CURRENT(COUNT$3)COUNT$4)=BIG_NUM;
```

```
      ELSE
      LEARN(COUNT$3+COUNT$ )=BIG_NUM;
       LEARN$DATA(LARGEST)=0;
      END;
      END;
     COUNT$4=COUNT$4+MARK$_2;
/*SHIFT TEST BIT ,IE TOGGLE CP IN*/
   CALL CP$IN$CLK;
   NODE$NO=NODE$NO+1;
/*CHECK IF TEST NODE ALREADY EXIST*/
    IF CUR_FLAG=TRUE THEN
   DO;
    DO COUNT=0 TO (SIZE-1);
      IF CURRENT(COUNT)=NODE$NO THEN
      DO;
       CALL CP$IN$CLK;
        NODE$NO=NODE$NO+1;
      END;
       END;
     END;
      ELSE
     DO;
   DO COUNT=0 TO (SIZE-1);
     IF LEARN(COUNT)=NODE$NO THEN
     DO;
      CALL CP$IN$CLK;
       NODE$NO=NODE$NO+1;
     END;
     END;
    END;
```

```
    END; /*DO WHILE*/
  END GO$NOGO$TEST;

/* SHORT_ALGO COMPARES THE 'LEARN' ARRAY DATA REPRESENTING THE MODEL
    WIRING PATTERN WITH DATA IN THE 'CURRENT' ARRAY REPRESENTING THE
    WIRING CONFIGURATION OF THE LOOM UNDER TEST. ANY DIFFERENCES ARE
    DECODED & STORED AS SHORT-CIRCIUT NODE NUMBERS IN THE 'SHORT' ARRAY*/
SHORT_ALGO:
 PROCEDURE;
K=SIZE-1;
 COUNT$S=0;
DO WHILE K>=1;
  CURRENT$STOP=K;
 I=K;
  CHECK=FALSE;
  IF CURRENT(K)<CURRENT(K-1) THEN
    DO;
    DO J=0 TO (SIZE-1);
      IF LEARN(J)=CURRENT(I) THEN
      DO;
        LEARN$STOP=J;
       L=J;
      END;
     END;
     DO WHILE(LEARN(L)<LEARN(L-1)) AND (L<>0);
       L=L-1;
     END;
      LEARN$START=L;
     DO WHILE (CURRENT(I)<CURRENT(I-1)) AND (I<>0);
```

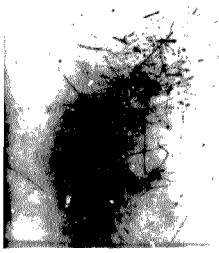
```
    I=I-1;
  END;
  CURRENT$START=I;
  DO COUNT=CURRENT$START TO CURRENT$STOP;
    CHECK=FALSE;
   DO COUNT$3=LEARN$START TO LEARN$STOP;
     IF CURRENT(COUNT)=LEARN(COUNT$3) THEN
      CHECK=TRUE;
    END;
    IF CHECK=FALSE THEN
      DO;
       SHORT(COUNT$5)=CURRENT(CURRENT$STOP);
        COUNT$5=COUNT$5+1;
       SHORT(COUNT$5)=CURRENT(COUNT);
        COUNT$5=COUNT$5+1;
       END;
   END;
   IF CURRENT$START=0 THEN
     K=0;
   ELSE
    K=CURRENT$START-1;
    END; /*IF DO*/
   ELSE K=I-1;
    END; /*DO WHILE*/
   IF COUNT$5>0 THEN
     PFD=TRUE;
 END SHORT_ALGO;

/* OPEN_ALGO COMPARES THE 'LEARN' ARRAY DATA REPRESENTING THE MODEL
   WIRING PATTERN WITH THE DATA IN THE 'CURRENT' ARRAY REPRESENTING
```

```
   THE WIRING PATTERN OF THE CURRENT LOOM UNDER TEST. ANY DIFFERENCES
    ARE DECODED & STORED AS OPEN-CIRCUIT NODE NUMBERS IN THE'OPEN' ARRAY*/
OPEN_ALGO:
 PROCEDURE;
 COUNT$6=0;
  DO K=0 TO (SIZE-1);
   CURRENT$STOP=K;
    I=K;
   CHECK=FALSE;
    DO J=0 TO (SIZE-1);
     IF LEARN(J)=CURRENT(I) THEN
       DO;
       LEARN$STOP=J;
        L=J;
      END;
     END;
     IF LEARN(L+1)>LEARN(L) THEN
       DO;
       LEARN$START=L;
       DO WHILE ((LEARN(L)<LEARN(L-1)) AND (L<>0));
         L=L-1;
        LEARN$START=L;
        END;
       CURRENT$START=I;
       DO WHILE ((CURRENT(I)<CURRENT(I-1)) AND (I<>0));
         I=I-1;
        CURRENT$START=I;
        END;
       DO COUNT=LEARN$START TO LEARN$STOP;
        CHECK=FALSE;
```

```
      DO COUNT$3 = CURRENT$START TO CURRENT$STOP;
        IF LEARN(COUNT)=CURRENT(COUNT$3) THEN
         CHECK=TRUE;
        END;
        IF NOT(BOOLEAN(CHECK)) THEN
          DO;
          OPEN(COUNT$6)=LEARN(LEARN$STOP);
           COUNT$6=COUNT$6+1;
          OPEN(COUNT$6)=LEARN(COUNT);
           COUNT$6=COUNT$6+1;
         END;
        END;
      END;
     END;
    IF COUNT$6>0 THEN
     RED=TRUE;
   END OPEN_ALGO;

/*MAIN INITIALISATION OF VARIABLES*/
    INIT:/*TURN ALL LEDS OFF*/
     P1=17H;
    P1=P1 AND 0CFH;
     P1=07H;
    P1=P1 OR 10H;
     CAP=0;
    INFO(3)=82H;
     J=1;
    INFO(2)=30H;
     LO_FLO=FALSE;
```

```
      P1=15H;
       P1=P1 AND 0EFH;
      P1=07H;
       P1=P1 OR 10H;
      PORT_0=07H;
       PORT_2=0FH;
      TCON=40H;
      TMOD=20H;
      TH1=0C9H;
       INT=90H;
      SCON=40H;
/* MAIN PROGRAM LOOP */
    LOOP:
      DO FOREVER;
      CALL TEST_SW;
       CALL S_TEST_SW;
      CALL LOAD_SW;
       CALL PROG_SW;
      IF (PROG=07H) AND (LD_FLG=TRUE) THEN
        DO;
        CALL RED_OFF;
         CALL GREEN_OFF;
        CALL YELLOW_ON;
         CALL PROG_EPROM;
        IF N_VERIFY=FALSE THEN
          CALL GREEN_ON;
        ELSE
          CALL RED_ON;
        CALL YELLOW_OFF;
        END;
```

```
IF LOAD=0B$ THEN
  DO$
  LD_FLG=TRUE$
   CAP=0$
  CUR_FLAG=FALSE$
   CALL RED_OFF$
  CALL GREEN_OFF$
   CALL YELLOW_ON$
  CALL APP_INIT$
   CALL GO$NGGO$TEST$
  DO I=0 TO LAST(S_DATA)$
    SBUF=S_DATA(I)$
   CALL DEL$
    CALL DEL_1$
  END$
   COUNT=0$
  I=0$
  DO WHILE I<(SIZE-1)$
    IF LEARN(I)>LEARN(I+1) THEN
    DO$
     IF COUNT=0 THEN
       STAR=I$
     COUNT=1$
      I=I+1$
    END$
   IF (LEARN(I)<LEARN(I+1)) AND (COUNT=1) THEN
     DO$
     FINI=I$
      CALL SPACE$
```

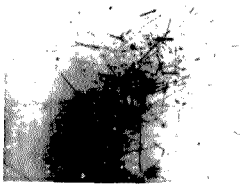


```
DO I=STAR TO FINI;
            REC_NO=LEARN(I);
           CALL ASC_NO;
            DO K=0 TO 1;
             SBUF=A_NO(K);
              CALL DEL;
             CALL DEL_1;
            END;
           SBUF=20H;
            CALL DEL;
           CALL DEL_1;
           END;
          SBUF=0DH;
           CALL DEL;
          CALL DEL_1;
           COUNT=0;
        END;
        IF (LEARN(I)<LEARN(I+1)) AND (COUNT=0) THEN
          I=I+1;
      END; /*DO WHILE*/
      CALL YELLOW_OFF;
    END;
     IF TEST=0EH THEN
     DO;
      CUR_FLAG=TRUE;
       CALL RED_OFF;
      CALL GREEN_OFF;
       CALL YELLOW_ON;
      RED=FALSE;
       CALL EPROM_LOAD;
```

```
CALL APP_INIT;
  CALL GO$NOGO$TEST;
 DO COUNT=0 TO (SIZE-1);
   SHORT(COUNT)=0;
  OPEN(COUNT)=0;
  END;
 CALL OPEN_ALGO;
  CALL SHORT_ALGO;
 DO I=0 TO LAST(S_DATA);
   SBUF=S_DATA(I);
  CALL DEL;
   CALL DEL_1;
 END;
 CALL MESSAG_OUT(.SHORT,LAST(SHORT));
  IF COUNT$S>0 THEN
  DO;
    CALL SPACE;
   REC_NO=(COUNT$S/2);
    CALL ASC_NO;
   DO K=0 TO 1;
     SBUF=A_NO(K);
    CALL DEL;
     CALL DEL_1;
   END;
    CALL FAULT;
  END;
  ELSE
  CALL PASS;
  DO I=0 TO LAST(O_DATA);
   SBUF=O_DATA(I);
```

```
    CALL DEL;
    CALL DEL_1;
    END;
  CALL MESSAG_OUT(.OPEN,LAST(OPEN));
    IF COUNT$6>0 THEN
    DO;
      CALL SPACE;
     HEC_NO=(COUNT$6/2);
      CALL ASC_NO;
     DO K=0 TO 1;
       SBUF=A_NO(K);
      CALL DEL;
       CALL DEL_1;
     END;
      CALL FAULT;
    END;
    ELSE
    CALL PASS;
    CALL YELLOW_OFF;
   IF RED=TRUE THEN
     CALL RED_ON;
   ELSE
     CALL GREEN_ON;
 END;
 IF S_TEST=0DH THEN
   DO;
    CALL RED_OFF;
     CALL GREEN_OFF;
    CALL YELLOW_ON;
     RED=FALSE;
```

```
CLR_FLAG=TRUE;
 DO I=0 TO (SIZE-1);
 LEARN(I)=CHG_SHORT(I);
  SHORT(I)=0;
END;
 CALL APP_INIT;
CALL GO$NOGC$TEST;
 CALL SHORT_ALGO;
DO I=0 TO LAST(S_DATA);
  SBUF=S_DATA(I);
 CALL DEL;
  CALL DEL_1;
END;
CALL MESSAGE_OUT(.SHORT,LAST(SHORT));
 S_TEST=0FFH;
CALL ONE_S_DEL;
 CALL ONE_S_DEL;
DO WHILE S_TEST<>0DH;
  CALL S_TEST_SW;
 CALL S_TEST_SW;
  CALL YELLOW_ON;
 DO J=1 TO 10;
   CALL TIME(100);
 END;
  CALL YELLOW_OFF;
 DO J=1 TO 10;
   CALL TIME(100);
 END;
 END;
```



```
    CALL YELLOW_ON;
     DO I=0 TO (SIZE-1);
      LEARN(I)=CH0_OPEN(I);
       OPEN(I)=0;
     END;
      CALL APP_INIT;
     CALL GO$NOGO$TEST;
      CALL OPEN_ALGO;
     DO I=0 TO LAST(O_DATA);
       SBUF=O_DATA(I);
      CALL DEL;
       CALL DEL_1;
     END;
     CALL MESSAGE_OUT(.OPEN,LAST(OPEN));
      CALL YELLOW_OFF;
     IF RED=TRUE THEN
       CALL RED_ON;
     ELSE
       CALL GREEN_ON;
   END;
  END; /*FOREVER*/
END START;
```

APPENDIX D

ILLUSTRATION OF PROTOTYPE LOOM TESTER