1985

Door opener MK1

Peter C. Arnold

University of Wollongong, parnold@uow.edu.au

Recommended Citation
http://ro.uow.edu.au/compsciwp/11
DOOR OPENER MK1

Peter Arnold

CSCI321 Project 1984
Department of Computing Science
University of Wollongong

Preprint No. 85-1
January 1985

P.O. Box 1144, WOLLONGONG N.S.W. 2500, AUSTRALIA
tel (042)-270-859
telex AA29022
I would like to thank Phillip McKerrow for his help and guidance in the main design concepts of this project and Michael Milway for his help in component procurement and design verification.

October 15, 1984
CONTENTS

<table>
<thead>
<tr>
<th>TITLE</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0 INTRODUCTION</td>
<td>4</td>
</tr>
<tr>
<td>2.0 SYSTEM CONFIGURATION</td>
<td>5</td>
</tr>
<tr>
<td>2.1 THE DOOR LOCKING MECHANISM</td>
<td>6</td>
</tr>
<tr>
<td>2.2 ADVANTAGES AND DISADVANTAGES</td>
<td>7</td>
</tr>
<tr>
<td>2.2.1 ADVANTAGES</td>
<td>7</td>
</tr>
<tr>
<td>2.2.2 DISADVANTAGES</td>
<td>8</td>
</tr>
<tr>
<td>3.0 OPERATION</td>
<td>9</td>
</tr>
<tr>
<td>3.1 HOST / CONTROLLER CONNECTION</td>
<td>9</td>
</tr>
<tr>
<td>3.2 MODES OF CONTROLLER OPERATION</td>
<td>9</td>
</tr>
<tr>
<td>3.3 COMMUNICATIONS FROM HOST TO CONTROLLER</td>
<td>10</td>
</tr>
<tr>
<td>3.4 COMMUNICATIONS CONTROLLER TO HOST</td>
<td>11</td>
</tr>
<tr>
<td>3.5 SAMPLE C PROGRAM</td>
<td>12</td>
</tr>
<tr>
<td>4.0 CONTROLLER SOFTWARE</td>
<td>13</td>
</tr>
<tr>
<td>4.1 SOFTWARE OVERVIEW</td>
<td>13</td>
</tr>
<tr>
<td>4.2 DATA AND PROGRAM STRUCTURES</td>
<td>13</td>
</tr>
<tr>
<td>4.2.1 RS232 BUFFERS</td>
<td>13</td>
</tr>
<tr>
<td>4.2.2 INPUT BUFFERS AND RS232 CHANNELS</td>
<td>14</td>
</tr>
<tr>
<td>4.2.3 COMMAND INTERPRETERS</td>
<td>15</td>
</tr>
<tr>
<td>4.2.4 DISPLAY BUFFERS</td>
<td>15</td>
</tr>
<tr>
<td>4.2.5 DISPLAY DRIVING ROUTINE</td>
<td>16</td>
</tr>
<tr>
<td>4.2.6 KEYBOARD AND KEYBOARD CHANNELS</td>
<td>16</td>
</tr>
<tr>
<td>4.2.7 MISCELLANEOUS ROUTINES</td>
<td>17</td>
</tr>
<tr>
<td>5.0 HARDWARE</td>
<td>18</td>
</tr>
<tr>
<td>5.1 BOARD CORRECTIONS</td>
<td>18</td>
</tr>
<tr>
<td>5.2 LIGHT EMITTING DIODE DEBUGGING AID</td>
<td>19</td>
</tr>
<tr>
<td>5.3 PROTOTYPE EPROM</td>
<td>19</td>
</tr>
<tr>
<td>6.0 CONCLUSION</td>
<td>20</td>
</tr>
<tr>
<td>APPENDIX A CIRCUIT DIAGRAMS</td>
<td></td>
</tr>
<tr>
<td>APPENDIX B PRINTED CIRCUIT BOARD ARTWORK</td>
<td></td>
</tr>
<tr>
<td>PRINTED CIRCUIT BOARD COMPONENT OVERLAYS</td>
<td></td>
</tr>
<tr>
<td>APPENDIX C SAMPLE C PROGRAM</td>
<td></td>
</tr>
<tr>
<td>APPENDIX D PROTOTYPE SOURCE LISTING</td>
<td></td>
</tr>
<tr>
<td>APPENDIX E PARTS LIST (+ SUGGESTED SUPPLIERS)</td>
<td></td>
</tr>
</tbody>
</table>

October 15, 1984
1. INTRODUCTION

This document is a report on the development of a password-keyboard security locking system designed to interface with up to two host computers for the purpose of password controlled access to computer facilities. It was designed for incorporation into the computing facilities of the Computing Science Department of Wollongong University particularly to provide a means of access to their machines by students after hours.

The system consists of a controller and one or two keyboard/display modules. It interfaces with the host machines via RS232 lines and the host machines determine whether access is to be granted. The host machines see the controller as being a dumb terminal. Users type their login ID and password on the keyboards, which are placed near entry points to the facilities, after appropriate prompts from the hosts. The hosts verify the passwords and, if valid, grant access to the user and tell the controller to unlock the door. The host machines then record to whom access was granted and when the doors open and close. People with door keys need not use the system and can enter the facilities using their keys.

Contained in this document are descriptions of how the unit works, how it should be used, construction details and design criteria containing a discussion on its relative merits and disadvantages.

October 15, 1984
2. SYSTEM CONFIGURATION

The system developed consists of several main parts, the most important of which is the controller. This is a microprocessor based board which interfaces between the various other components of the system. It determines all the timing of various events. It scans the keyboards, updates the displays, sends and receives data from the host machines, reads various sensors and drives the door locking mechanism.

Up to two keyboards can be connected to the controller, along with their respective displays. The keyboards used in the development of the prototype were AIM65 keyboards primarily because they were cheap and available, although thin membrane keyboards would be better propositions for working systems since they are likely to be cheaper and more durable in an external environment. The displays used were 16 character alphanumeric light emitting diode units. Here again better displays would be liquid crystal units due to the fact that they are off the shelf items and their price is much less than the led displays used.

The controller interfaces with the host machines via RS232 lines utilising ASCII codes. These lines are user settable to most baud rates and a variety of bit code formats. Door unlocking is enabled by the connection of an Electric Door Latch to the controller and sensing whether the door is open or shut is accomplished by a magnetic reed switch assembly placed in the door and connected to the controller.

A typical configuration is shown in fig 2.0. The controller and one keyboard/display module are placed inside the door to be monitored. The second keyboard and display is placed outside the door. Operation is as follows:

TYPICAL

[1] One host machine sends the message "login:" to both displays.

[2] A user approaches the external keyboard/display and selects which host he wants to gain access through.

[3] The user then types his login ID. (This is echoed by the host)

[4] The host checks the login and, if valid, prompts "password:

October 15, 1984
SYSTEM CONFIGURATION

FIG 2.0
[5] The user then types his password (This is not echoed by the host)

[6] Upon verification that the password matches that for the login ID given, the host sends the message "open" and instructs the controller to unlock the door.

[7] Four seconds after displaying "open" on the display, the electric door latch is energised for a further four seconds, allowing access through the door.

[8] The user opens the door and walks through, meanwhile a sensor causes the controller to report to the hosts that the door has opened.

[9] The user closes the door and the sensor causes the controller to report to the host that the door is closed.

[10] The user exits the facility using the same steps described previously except that the internal keyboard is used.

In other words as users enter and exit the facility the host machines record to whom it has given access, through which keyboard, and at what time. They also record when the door is opened, when the door is closed and if the door has been left open for more than a minute in which case a warning may be broadcast to all users connected to the host. Reports on the door state are provided automatically by the controller as is the timing for the door opening process.

The internal keyboard, though not essential, is useful to monitor exit from the facility. Although it is possible to exit by using the door handle (from the inside) it is still important to monitor who is inside and who has exited the facility for security purposes. In this sense it is important to impress upon potential users their liability if they are recorded as entering the facility and not exiting it. They are "officially" inside the building and are accountable if any damage occurs or equipment goes missing while they are there.

2.1. THE DOOR LOCKING MECHANISM

The normal door locking mechanism consists of an external key opening device, an internal handle, the bolt or tongue and the strike mechanism. The bolt is the extendible metal protrusion which juts out of the middle of the door edge alongside the handle. It is the part retracted when the handle is twisted. The strike is the (usually) immobile metal or wood retainer which encloses the bolt and holds the door closed.

The electric door strike used in the prototype retracts
the striker plate when energised freeing the door to move, when the strike is not energised the bolt is firmly enclosed in the strike mechanism as in the usual closed door situation. The door can still be opened from the outside using a key as before or from the inside using the handle via the conventional bolt retraction scheme. This configuration permits external entry to key-holders at all times and allows exit from inside at any time by anyone no-matter whether the controller or hosts are working or otherwise. This is an important safety consideration in the case of power failure, host or controller crashes or fire.

2.2. ADVANTAGES and DISADVANTAGES

The following notes concern the relative advantages and disadvantages of this type of security configuration.

2.2.1. ADVANTAGES

[1] Passwords can be used which can be modified or changed by individual users at regular intervals if they feel that their password is not safe.

[2] Records can be kept on who entered the facility, at what time they entered and at what time they left.

[3] Warning messages can be broadcast to facility users via the host if the door is left open for more than a predetermined period (currently set at one minute).

[4] Users, because there is a record of their presence in the facility are forced to take on the responsibility of helping to protect it.

[5] The system is relatively cheap to build.

[6] There is no need to issue keys to anyone except staff personnel and hence there is no great problem in keeping track of the keys that are issued and no need to change locks regularly to prevent key copiers too easy an access.

[7] Key-holders can bypass the system

[8] People inside the facility can exit at any time using the door handle as usual in the event of power failure or emergency.

[9] The system can be built in-house and hence customised to meet particular specifications.

October 15, 1984
In the case of a two-host system, if one host goes down, users can still access the facility using the other host. If both hosts go down those inside can get out and those outside with keys can get in. (Anyone without a key would not gain entry in this situation - but then again who would want to get in if the facilities are down anyway).

Allows unsupervised access to facilities for users at any time after hours. (Subject of course to the system supervisors jurisdiction since he/she may program the hosts at will to allow either a select group or anyone in and during any selected time period.)

Reduces the need for paid after-hour supervision and hence saves money.

2.2.2. DISADVANTAGES

This system only keeps honest people out. Anyone who really wants to get in won't spend hours trying to guess the password - they will break a window or something.

The system depends on the integrity of those allowed in to not allow others in etc. And it also depends on the fear of accountability of users to be used properly. In this sense IT IS IMPORTANT TO MAKE PENALTIES FOR SYSTEM ABUSE HARSH AND EVEN MORE IMPORTANT TO PUBLICISE THIS FACT.

The prototype developed has a few faults, these being the susceptibility of the keyboard to weather and the price of the LED display. The current scheme has over $100 of electronics exposed to the outside - to weather and vandals. This problem could be reduced by intelligent mounting of both the keyboard and display and by getting cheaper devices. E.g. a cheaper and more weather-proof keyboard may be found in a thin membrane keyboard such as made by CHERRY. A cheaper display such as the liquid crystal displays sold by PROMARK or DANEVA (displays made by SHARP) would be a better proposition for cost as well as the fact that (allowing for availability) the device is off the shelf whereas the LED device requires a bit of construction.

The system is not off the shelf and requires a bit of building and assembly.

October 15, 1984
3. OPERATION

The following text describes how the controller and host interface, how to set up the controller for use. It also includes a description of the sample C text in APPENDIX C.

3.1. HOST / CONTROLLER CONNECTION

The controller connections are shown in FIG 3.0. They include 16 pin dip sockets for the two keyboards and two displays – note that if desired only one keyboard/display need be connected and the controller will still work without modification. Also connection points for the electric door strike mechanism, the magnetic reed switch, the RS232 sockets, the reset switch (if required), the power transistor and the five volt regulator are shown on this diagram.

FIG 3.0 includes a table listing the various switch settings and their consequences. SW1 to SW4 control the baud rate at which data is sent and received by the controller. SW5 to SW7 determine the data envelope which is sent and received by the controller. SW8 determines whether the controller will operate in TERMINAL mode or NORMAL mode. SW1 to SW7 are read at power up or reset and are set from that point on. To change the settings the switches must be changed and a power up or reset must be initiated. SW8 can be changed at any time and the controller need not be reset etc. to change modes.

3.2. MODES OF CONTROLLER OPERATION

The controller may operate in two modes. The first mode is TERMINAL mode which is selected by moving the dip switch No. 8 to the one (1) position. In this mode the controller transmits characters from either keyboard without a preceding identifier character (i.e. 1 or 2). Data transmitted from the host machines may go to any display (actually the display last written to by that particular host) and both upper and lower case characters are displayed. No command characters are recognised in this mode. The reason for including this mode in the controller was to allow it to "log in" to the host computers for development and demonstration purposes.

The second mode is NORMAL mode which is the normal mode for the controller to operate under when it is policing door entry. This mode is selected by moving the dip switch No. 8 to the zero (0) position. Movement between modes may be done at any time and does not require re-initialisation of the controller. All characters sent from the keyboards in this mode are preceded by an identifier character, i.e. characters sent from keyboard 1 are preceded by a '1' and characters sent from keyboard 2 are preceded by a '2'.

October 15, 1984
1.2. COMMUNICATIONS FROM HOST TO CONTROLLER

The host computers communicate to the controller using a number of commands and a few conventions. The main conventions are as follows:

1) All upper case characters sent to the controller are interpreted by it as commands.

2) All lower case letters sent to the controller are sent by it to the displays.

3) To direct data to display 1 the data should be preceded by "\1".

4) To direct data to display 2 the data should be preceded by "\2".

5) To print a backslash two backslash characters are required e.g. "\\".

6) The controller recognises only two control characters. "bs" backspace (HEX 08) and "cr" carriage return (HEX OD)

The commands and their actions are as follows:

S - request checksum from controller. This command causes the controller to do a checksum of its internal ROM and report the results. It is useful for testing whether the controller ROM is uncorrupted and in fact whether the controller is working properly. The controller will respond with a "S<cr>nn" where S stands for checksum <cr> stands for carriage return n stands for a numeric character.

I - initialise controller. This causes the controller to do a software reset. It initialises all buffers to empty and causes the normal controller power up routines to execute.

D - open door. Once this command is issued, the controller will wait for four seconds and then unlock the door for four seconds.

R - report door status. When the controller receives this command it immediately checks the status of the door. If the door is closed the controller sends a "c" character. If the door is open a "o" character is sent.
3.4. COMMUNICATIONS CONTROLLER TO HOST

Communications from the controller to the host machines also follow conventions. These conventions are as follows:

1) All characters sent from keyboard 1 are preceded by "1". e.g. if you typed "pwa<cr>" it would be sent to the host as "1pwlw1al<cr>"

2) All characters sent from keyboard 2 are preceded by "2". e.g. if you typed "pwa<cr>" it would be sent to the host as "2p2w2a2<cr>"

3) All characters sent from the controller without a preceding identifier number should be interpreted by the host as report characters. A report character tells the host computer something about either the state of the door or the state of the controller. Report characters are followed immediately by a <cr> character.

4) When a report character is sent it is sent to both host machines.

5) Keyboard typed characters are sent only to the host selected by the user (the typer).

The following report characters are currently defined in the controller:

S<cr> - checksum. This is the response to the host's request for a checksum and is immediately followed by two ASCII characters which represent the checksum.

P<cr> - checksum on power up. This report is generated when the controller is powered up, when it is reset either by its reset button or from the host via an I command. It is immediately followed by two ASCII characters which represent the checksum.

C<cr> - door has closed.

O<cr> - door has opened

c<cr> - door is closed. This response will occur if an R command is received from the host and the door is closed.

o<cr> - door is open. This response will occur if an R command is received from the host and the door is open.
FIG 3.1 EXAMPLE C PROGRAM STATE DIAGRAM

- **STATE 0**
- **PRINT "LOGIN"**
- **PRINT "ERROR,"**

- **VALID LOGIN ID**
- **CR**
- **INVALID PASSWORD**

- **STATE 1**

- **VALID PASSWORD**

- **SEND OPEN DOOR COMMAND**

- **PRINT "OPEN,\""**
3.5. **SAMPLE C PROGRAM**

The sample C of APPENDIX C program operates as shown in FIG 3.1. It assumes the password system of the host consists of a "login identifier" (or login ID) for individual users, which may be known by anyone, and a password which is known only to the individual and is linked in an encrypted form with the login ID in a file "/etc/passwd" such as implemented in UNIX.

When the controller sends characters these are buffered. There are individual buffers for keyboard 1, keyboard 2 and reports. When a carriage return character is received from the controller the buffer to which the <cr> belongs is processed.

There are two main states in which the C program can be and these states occur for both keyboards individually. State 0 is the "login" state. In this state the C program requests a login ID. State 1 is entered after a valid login ID has been received when in state 0. Empty lines (only a <cr> character) cause the issuance of the prompt "login:" and entry to state 0. Invalid login ID's result in an "error, login:" message being displayed and entry to state zero.

In state 1 the program can progress in three paths. An empty line will cause a "login:" prompt and entry to state 0. An invalid password (i.e. one which does not match the login ID password) causes "error, login:" to be prompted and enters state 0. A valid password will result in the prompt "open, login:" and will send a "D" - open door command to the controller and will cause re-entry to state 0.

Reports for warnings, checksums on power-up, door opening and closings are all recorded in a file "logrecord". The report messages are long for demonstrative purposes and should be abbreviated in practical systems for efficient data storage. The form of a report entry is:

```
  date/time of report
  report
```

This demonstration routine was run by putting the controller into TERMINAL mode, logging into the host machines into the directories where the routine existed, and then executing the routines, one for each host. Once the routines were running the controller was switched to NORMAL mode and used as described in section 2.0. Each host keeps records of the users it has admitted to the facility and both hosts record the reports generated by the controller.

October 15, 1984
4. **CONTROLLER SOFTWARE**

The following notes give a brief description of the essential components of the controller software. FIG 4.0 contains a data path chart which illustrates the various possible data paths in the controller and some of the routines that transfer and manipulate that data.

4.1. **SOFTWARE OVERVIEW**

Two "types" of routine are used by the controller. The first type are the routines which service interrupts which may occur at any time. All interrupts have the same priority although a few relatively long interrupt service routines will allow themselves to be interrupted. Interrupt routines service the RS232 transmitters and receivers and various timers.

The other type of routines are those which the main program runs one after the other. These include keyboard reading routines, display updating routines, command interpreters and various routines which move data from one buffer to the next. They are relatively low level tasks and are not time critical. They all yield priority to interrupts.

4.2. **DATA AND PROGRAM STRUCTURES**

Data structures in the controller consist mainly of 256 byte buffers used for RS232 input and output and buffers used for display or command data. It is best to consider the controller from the RS232 receiver/transmitters and work down from there.

4.2.1. **RS232 BUFFERS**

There are two RS232 transmitter/receivers called ACIA's (from this point onwards referred to as ACIA-A and ACIA-B) one of which is connected to the first host computer, the second of which to the second. Each ACIA requires two buffers - one into which is put data received from the host machine and one from which to draw data to send to the host. These buffers are called:

- **RIABUF** - RS232 input buffer for ACIA-A. (Data from host A is put here)
- **RIBBUF** - RS232 input buffer for ACIA-B. (Data from host B is put here)
- **ROABUF** - RS232 output buffer for ACIA-A. (Data to be sent to host A is placed here)

October 15, 1984
FIG 4.0 DATA PATHS
ROBBUF- RS232 output buffer for ACIA-B. (Data to be sent to host B is placed here)

The ACIA's are serviced by interrupt routines. When the output buffers are empty the transmitters are shut down for one millisecond (timed by 6522 timers) after which the appropriate buffers are checked again. The routines which service the ACIA's are:

RSATCH- transmit characters through ACIA-A to host-A from ROABUF

RSBTCH- transmit characters through ACIA-B to host-B from ROBBUF

RSARCH- receive characters from ACIA-A (from host-A) and buffer in RIABUF

RSBRCH- receive characters from ACIA-B (from host-B) and buffer in RIBBUF

TBINT- when ROBBUF is empty timer B is started to count for one millisecond and the transmitter in ACIA-B is shut down. At timer B timeout TBINT is called. TBINT then checks ROBBUF to see if there are any characters to send. If there are then the transmitter in ACIA-B is restarted, if not timer B is restarted.

TAINT- when ROABUF is empty timer A is started to count for one millisecond and the transmitter in ACIA-A is shut down. At timer A timeout TBINT is called. TAINT then checks ROABUF to see if there are any characters to send. If there are then the transmitter in ACIA-A is restarted, if not timer A is restarted.

4.2.2. INPUT BUFFERS AND RS232 CHANNELS

Data from RIABUF and RIBBUF are channeled to one of two input buffers, depending on which display the host is currently addressing, by a routine called

TRII - Transfer RS232 Input buffer data to Input buffer.

This routine reads data from a RS232 input buffer and vectors it through the appropriate channel to the appropriate 256 byte input buffer. These channels and input buffers are:

CHAN1- channel for RIABUF. If this channel contains '1' then data is moved from RIABUF to I1BUF. If this channel = '2' then data is moved from RIABUF to I2BUF.

October 15, 1984
Fig. 4.0  DATA PATHS
ROBBUF— RS232 output buffer for ACIA-B. (Data to be sent to host B is placed here)

The ACIA's are serviced by interrupt routines. When the output buffers are empty the transmitters are shut down for one millisecond (timed by 6522 timers) after which the appropriate buffers are checked again. The routines which service the ACIA's are:

RSATCH— transmit characters through ACIA-A to host-A from ROABUF

RSBTCH— transmit characters through ACIA-B to host-B from ROBBUF

RSARCH— receive characters from ACIA-A (from host-A) and buffer in RIABUF

RSBRCH— receive characters from ACIA-B (from host-B) and buffer in RIBBUF

TBINT— when ROBBUF is empty timer B is started to count for one millisecond and the transmitter in ACIA-B is shut down. At timer B timeout TBINT is called. TBINT then checks ROBBUF to see if there are any characters to send. If there are then the transmitter in ACIA-B is restarted, if not timer B is restarted.

TAINT— when ROABUF is empty timer A is started to count for one millisecond and the transmitter in ACIA-A is shut down. At timer A timeout TBINT is called. TAINT then checks ROABUF to see if there are any characters to send. If there are then the transmitter in ACIA-A is restarted, if not timer A is restarted.

4.2.2. INPUT BUFFERS AND RS232 CHANNELS

Data from RIABUF and RIBBUF are channeled to one of two input buffers, depending on which display the host is currently addressing, by a routine called

TRII - Transfer RS232 Input buffer data to Input buffer.

This routine reads data from a RS232 input buffer and vectors it through the appropriate channel to the appropriate 256 byte input buffer. These channels and input buffers are:

CHANA— channel for RIABUF. If this channel contains '1' then data is moved from RIABUF to IIBUF. If this channel = '2' then data is moved from RIABUF to I2BUF.
CHANB - channel for RIBBUF. If this channel contains '1' then data is moved from RIBBUF to I1BUF. If this channel = '2' then data is moved from RIBBUF to I2BUF.

I1BUF - input buffer 1. Data from this buffer is sent to display 1 in TERMINAL mode. In NORMAL mode lower case data is displayed and upper case data is interpreted as commands.

I2BUF - input buffer 2. Data from this buffer is sent to display 2 in TERMINAL mode. In NORMAL mode lower case data is displayed and upper case data is interpreted as commands.

The channels are changed if the data in one of the RS232 input buffers is "1" or "2". These character sequences sent by the host cause the data following them to be sent to "1" - I1BUF or "2" - I2BUF.

4.2.2. COMMAND INTERPRETERS

There are two command interpreters. They are

COM - NORMAL mode command interpreter. COM takes data from I1BUF and I2BUF sending lower case characters to DISPl and DISP2 which are the display buffers for display 1 and display 2 respectively. It also interprets and executes any commands sent from the hosts. (note that lower case characters are converted to upper case to be displayed)

COMM - TERMINAL mode command interpreter. COMM merely transfers displayable characters from I1BUF and I2BUF to their appropriate display buffers. COMM also converts all lower case characters to upper case for displaying.

4.2.4. DISPLAY BUFFERS

The two display buffers are called:

DISPl - display buffer 1
DISP2 - display buffer 2.

They are each thirty two characters long although only sixteen characters from each buffer are displayed using the display modules developed for the prototype (thirty two characters were allowed for larger future displays). Characters are pushed onto the display buffers by calling the routine

October 15, 1984
DSPPB—push character onto display buffer

The prototype implementation uses the first two characters to display the host currently being accessed by that keyboard/display module (either "A" for host-A or "B" for host-B).

Characters are pushed onto the display buffer from character two (third character from left) to character fifteen (rightmost character of display). Once these spaces are filled the data is scrolled onto the display with each new character occupying character fifteen and character two being lost from the display. A carriage return clears characters two to fifteen. A backspace clears the last entry from the display.

4.2.5. DISPLAY DRIVING ROUTINE

This routine is called

DISPL—Update display (16 character LED module)

and writes the data from the display buffers DISPl and DISP2 to the display modules.

4.2.6. KEYBOARD AND KEYBOARD CHANNELS

The keyboards are scanned about once every five milliseconds. To read a valid new key press the same key must be read twice successively and this key must be different from the previously read key. If these conditions are satisfied the key is then examined. There is no n-key roll-over capability in the prototype design. The keyboard is scanned once and the first key closure is the one that is read. If a shift or control character is encountered another scan of the keyboard occurs which does not include the shift/control row.

If the key read is F1 then output from this keyboard is sent to host-A. If the key read is F2 then output from this keyboard is sent to host-B. If the key read is F3 then the host to which this keyboard sends data changes from one to the other. (i.e. if data was being sent to host-A it is now sent to host-B and visa-versa). The routines etc. used to read the keyboard and send their data to the appropriate output buffers are as follows:

KBDO—Key interpreter. Vectors data to ROABUF or ROBBUF via CHAN1 and CHAN2.

KBO—also programs CHAN1 and CHAN2 if characters f1, f2 or f3 are keyed in.

October 15, 1984
SCAN - scan keyboard. SCAN also looks up KTABLE for the appropriate ascii value for the key pressed.

KTABLE - table of ascii values for keys, ascii values are ordered by row and column.

CHAN1 - channel for keyboard 1. If CHAN1 contains HEX 0A then data from keyboard 1 is sent to ROABUF. If CHAN1 contains HEX 0b then data from keyboard 1 is sent to ROBBUF.

CHAN2 - channel for keyboard 2. If CHAN2 contains HEX 0A then data from keyboard 2 is sent to ROABUF. If CHAN2 contains HEX 0b then data from keyboard 2 is sent to ROBBUF.

4.2.7. MISCELLANEOUS Routines

The door opening and sensing routines are as follows:

TDINT - count down for wait time and door open time (interrupt driven).

ODOOR - start door opening procedure

DOORCB - door checking routine.

WARNING - check if door has been open for more than one minute.

Other miscellaneous routines are:

CHECKSUM - do a checksum of ROM and report the results

SET1 - set vectors for display 1, keyboard 1, RIABUF access. This routine is called before calling other routines (e.g. KBDO, DISP etc.) so that their variable pointers (or vectors) will be set correctly.

SET2 - set vectors for display 2, keyboard 2, RIBBUF access. This routine is called before calling other routines (e.g. KBDO, DISPB etc.) so that their variable pointers (or vectors) will be set correctly.

October 15, 1984
This chapter is addressed to anyone who wishes to construct the controller board. There are a few corrections which should be made to the controller board which became evident when the prototype was constructed. These corrections had been made to the circuit diagram at the time of writing but had not been added to the artwork labeled DOOR OPENER MK1. When they are incorporated into the artwork the artwork will be re-named DOOR OPENER MK1.1.

The design of the controller board was such that 6522 VIA chips were used where cheaper 6521 PIA chips in most cases would have sufficed. The reason for this is that the wiring was easier with all the I/O chips being the same and the enhancements of the VIA chips, such as timers and the like, allowed a greater deal of flexibility in the software that could be developed on the board. This board incidently is a very general purpose design which could be utilised in a number of applications.

A suggestion which could aid the prospective builder is to get the board made plated through since there are a number of interconnections between the upper and lower planes. The diagram FIG 4.0 gives a broad generalised lay out of the board which may simplify connections to it - a complete circuit diagram and board layout are shown in APPENDIX A and APPENDIX B respectively.

NOTE that the protection diode D5 is not placed on the board (perhaps another oversight) and it should be connected in the manor shown in the circuit diagram of APPENDIX A at the termination point where the connections of the ELECTRIC DOOR STRIKE mechanism to the controller enclosure are made. If it is not placed in the circuit and the door strike mechanism is inductive then the output transistor Q2 may not survive the high voltage developed by the mechanism when it is de-energised.

Note also that the two ACIA chips are around the opposite way to the rest of the chips. They may not like being put around the wrong way so TAKE CARE when inserting them to the printed circuit board.

5.1. BOARD CORRECTIONS

The following list is the summation of these corrections.

[1] ACIA B (IC8) should be provided with its own 1.8432 MHZ crystal across pins 6 and 7. The pin 6 to pin 6 connection from ACIA A to ACIA B (IC7 - IC8) should be broken.

October 15, 1984
[2] No provision has been made on the board for diodes D10 and D11, connected near the reset switch.

[3] The reset switch should be connected across the capacitor C1 not the resistor R3.

5.2. LIGHT EMITTING DIODE DEBUGGING AID

The six light emitting diodes near the dip switch were designed to be used as debugging aids particularly for interrupt servicing routines. When an interrupt occurs the LED which corresponds to that interrupt lights. All LEDs are turned off about every five milliseconds as the main program cycles through a routine called LOFF. The interrupts associated with each LED are noted in the circuit diagram of APPENDIX A.

5.2. PROTOTYPE EPROM

The eprom used in the prototype was a 2716. The circuit board was designed to house either this chip or a 2732. If a 2732 eprom is used a jumper connection must be changed on the board as shown in the circuit diagram of APPENDIX A. For eproms different to 2716 or 2732 standards, the printed circuit board may have to be changed.

October 15, 1984
6. CONCLUSION

If this door entry system is used properly it could provide a relatively cheap and reliable method of controlling access to system facilities after hours and without costly supervision. The prototype developed was built up to circuit board stage and at the time of writing had not been packaged and tested in a real life environment, but I am confident that it will work as specified and will be highly reliable if and when put to the test.

October 15, 1984
CIRCUIT DIAGRAM FOR LED DISPLAY MODULE
APPENDIX B

PRINTED CIRCUIT BOARD ARTWORK

PRINTED CIRCUIT BOARD COMPONENT OVERLAYS

October 15, 1984
DOOR OPENER
by PETER ARNOLD
MKI 1984
/* EXAMPLE door opening program */
/* by PETER ARNOLD 12/10/1984 */

#include <stdio.h>
#define MAXBUF 100

main ()
{

int forever = 1; /* keyboard 1 buffer */
char BUFF1[MAXBUF]; /* keyboard 1 buffer */
char BUFF2[MAXBUF]; /* keyboard 2 buffer */
int state1 = 0; /* state of keyboard 1 */
char salt1[MAXBUF]; /* password encryption for kbd 1 */
int state2 = 0; /* state of keyboard 2 */
char salt2[MAXBUF]; /* password encryption for kbd 2 */
char logl[MAXBUF]; /* login ID for keyboard 1 */
char log2[MAXBUF]; /* login ID for keyboard 2 */
int ptr1 = 0; /* BUFF1 index */
int ptr2 = 0; /* BUFF2 index */
int ptrr = 0; /* BUFFR index */
int com; /* command or character */
int c; /* character */

system ("stty -echo cbreak");
while (forever)
{
com = getchar();
if (((com <= '9')&&(com >= '0'))) com = com - '0';
if (((com == 1)||(com == 2))
{
/* valid input from 1 or 2 */
c = getchar();
if (com == '15') com = '\l2'; /* map cr to \n */
if (com == 1)
{
/* command from 1 */
if (state1 != 1)
{
printf ("\a", com);
if (((c >= 'A')&&(c <= 'Z')))
putchar (c - 'A' + 'a');
else putchar (c);
}
ptr1 = buffer (BUFF1,c,ptr1);
if (c == '\n')
state1 = COMMAND (state1,BUFF1,salt1,com,log1);
}
else
{
/* command from 2 */
if (state2 != 1)
{
printf ("\a", com);
if (((c >= 'A')&&(c <= 'Z')))
putchar (c - 'A' + 'a');
else putchar (c);
}
}
}

October 15, 1984
ptr2 = buffer(BUFF2, c, ptr2);
if (c == '\n')
    state2 = COMMAND(state2, BUFF2, salt2, com, log2);
}
else /* report command */
{
    ptrr = buffer(BUFFR, com, ptrr);
    if (com == '\n') REPORT(BUFFR);
}
}

buffer (BUFF, c, ptr)
char BUFF[];
int c;
int ptr;
{
    switch (c)
    {
    case '\n' : BUFF[ptr] = '\0';
        return(0);
        break;
    case '\b' : if (ptr != 0) ptr--;
        return(ptr);
        break;
    default : BUFF[ptr++] = c;
        return (ptr);
    }
}

COMMAND (state, BUFF, salt, chan, log)
int state;
char BUFF[];
char salt[];
int chan;
char log[];
{
    int length;
    int notfound;
    char line[MAXBUF];
    FILE *fp, *fopen();
    int i,j;

    if (BUFF[0] == '\0')  /* COMMAND = NEWLINE */
    {
        printf ("\%d\nlogin : ", chan);
        return (0);
    }
    else
    {
        if (state == 0)
        { /* LOGIN WORD IN BUFF */

October 15, 1984
- 3 -

```
length = strlen(BUFF);
fopen ("/etc/passwd","r");
notfound = 1;
while (notfound)
  { /* look for login ID */
    if (fgets (line,MAXBUF,fp))
      {
        i = 0;
        j = length;
        if (j < strlen (line))
          while (i < j )
            if (BUFF[i] != line[i]) j = i;
            else i++;
        if (i == length) notfound = 0;
      }
      else break; /* end of file */
    }
fclose(fp);
if (notfound)
  { /* NO LOG FOUND */
    printf("\%d
error, login:",chan);
    return (0);
  }
else
  { /* LOG FOUND, GET ENCRYPTION */
    i = length + 1;
    j = 0;
    while ((log[j] = BUFF[j])) j++;
    j = 0;
    while ((salt[j] = line[i++]) != ';') j++;
    salt[j] = '\0';
    printf("\%d
password : ",chan);
    return(1);
  }
else if (state == 1)
  { /* BUFF CONTAINS PASSWORD */
    if (strcmp(crypt(BUFF,salt),salt))
      { printf("\%d
error, login:",chan);
        return(0);
      }
    else
      { /* PASSWORD ACCEPTED */
        printf("\%d
open, login:D",chan);
        system("date >> logrecord");
        fp = fopen("logrecord","a");
        fprintf (fp,"access to %s from %d
",log,chan);
        fclose (fp);
        return(0);
      }
  }
else return(0); /* ERROR */
```

October 15, 1984
REPORT (BUFF)

char BUFF[];
{
    FILE *fp, *fopen();
    char sumlow, sumhi;

    switch (BUFF[0])
    {
        case 'O':
            system("date >> logrecord");
            fp = fopen("logrecord", "a");
            fprintf(fp,"door opened\n");
            fclose(fp);
            return(0);
            break;

        case 'C':
            system("date >> logrecord");
            fp = fopen("logrecord", "a");
            fprintf(fp,"door closed\n");
            fclose(fp);
            return(0);
            break;

        case 'P':
            system("date >> logrecord");
            fp = fopen("logrecord", "a");
            sumhi = getchar();
            sumlow = getchar();
            fprintf(fp,"Power up checksum = %c%c\n", sumhi, sumlow);
            fclose(fp);
            return(0);
            break;

        case 'W':
            system("date >> logrecord");
            fp = fopen("logrecord", "a");
            fprintf(fp,"WARNING - door left open beyond time limit\n");
            fclose(fp);
            return(0);
            break;

        default:
            system("date >> logrecord");
            fp = fopen("logrecord", "a");
            fprintf(fp,"ERROR character (%c) was detected\n",BUFF[0]);
            fclose(fp);
            return(0);
            break;
    }
}
SAMPLE OUTPUT FROM C PROGRAM <with comments>
Mon Oct 15 16:18:00 EST 1984
access to pwa from 1  < l stands for keyboard 1>
Mon Oct 15 16:18:33 EST 1984
door opened
Mon Oct 15 16:18:37 EST 1984
door closed
Mon Oct 15 16:19:04 EST 1984
access to pwa from 2  < 2 stands for keyboard 2>
Mon Oct 15 16:19:09 EST 1984
door opened
Mon Oct 15 16:20:10 EST 1984
WARNING - door left open beyond time limit
Mon Oct 15 16:21:00 EST 1984
door closed
Mon Oct 15 16:19:30 EST 1984
Power up checksum = 45  < controller has been reset>

October 15, 1984
RS232 A CHANNEL
EQU $41
$68
100 ;
ADDRESS
OFFSET FOR
WRITE POINTER
RS232 A OUTPUT READ POINTER

RS232 B INPUT
EQU
INPUT CHARACTER VECTOR
$5A
DEBOUNCED
EQU
* $BOOO
$4E ; ROW
EGU $94
I
WRITE
COUNTER
****
EQU
INPUT COLUMN.

ADDRESS PAGE VARIABLES

INPUT BUFFER VECTORS
EGU $9003
DSP2ADR EQU
SUM OF:B)
2
EQU
RS232
B
EGU
RS232 -)
RS232 B OUTPUT READ POINTER

KEYBOARD VARIABLES ****

CHAN1 EQU $4A
CHAN2 EQU $4B
COLADR EQU $4C
ROWADR EQU $4E
COLCHR EQU $50
DEBCHR EQU $52
OFFSET EQU $55
OLDCHR1 EQU $5A
DEBCHR1 EQU $5C
DEBCHR2 EQU $5E
ROW EQU $59
COL EQU $5E
CURRENT KEYBOARD ROW
CURRENT KEYBOARD COLUMN

BUFFER VARIABLES ****

ROABUT EQU $60
ROABB EQU $61
ROABB EQU $62
ROABB EQU $63
RIABUT EQU $64
RIABB EQU $65
RIABB EQU $66
RIABB EQU $67
ILI B EQU $68
ILI B EQU $69
ILI B EQU $70
ILI B EQU $71

RS232  -> INPUT ****
CHAN A EQU $80
CHAN B EQU $81
TASPEC EQU $82
TBSPEC EQU $83

DOOR AND PORTS ****

DISABLE EQU $90
WAITCNT EQU $91
WARNCNT EQU $92
SWITCH EQU $93
DOROLD EQU $94
DORB EQU $95
CHNUM EQU $96
WRNFLG EQU $98
HUND EQU $99

VECTORS ****

TIP EQU $A0
DSP EQU $A5
DSPW EQU $A7
DSPW EQU $A9
DSPD EQU $A7
DSPAD EQU $A8
TNUM EQU $A0
TNUM EQU $A5
T5PEC EQU $AF
CHN EQU $B1
RPOP EQU $B3

DISPLAY ***

DSP1ADR EQU $9000
DSP1ADR EQU $9001
DSP1ADR EQU $9002
DSP2ADR EQU $8000
DSP2ADR EQU $8001
DSP2ADR EQU $8002
DSP2ADR EQU $8003
DSP2ADR EQU $8004
// Source code for a microcontroller program

; Dynamic memory allocation

; Keyboards

; Column addresses

; Row addresses

; Command addresses

; Status registers

; Data registers

; Timer addresses

; Door addresses

; RS232 ports

; Data direction registers

; Display addresses

; Program main routine
INIT6

; INITIALISE DISPLAY WRITE POINTERS

; INITIALISE SPECIAL CHARACTER FLAG

; INITIALISE PORTS AND DOOR

; START KEYBOARD COUNTER

; INITIALISE DISPLAY

; CLEAR CURSORS IN DISPLAYS

; DO IT TWICE TO MAKE SURE

; CLEAR CHARACTER 'P'

; DO CHECK SUM SEND RESULTS

; GET CHARACTER FROM KEYBOARD

; NO KEY

; IS CHARACTER = F1?

; CHAN = A

; IS CHARACTER = F2?

; CHAN = 0B

; IS CHARACTER = F3?

; CHAN = A?

; IS CHANNEL = A?
PUSH CHARACTER ONTO B BUFFER

; PUSH CHARACTER ONTO A BUFFER

; PUSH TKBNUM ONTO OUTPUT BUFFER B

; PUSH CHARACTER ONTO RS232 OUTPUT BUFFER B

; PUSH CHARACTER ONTO RS232 OUTPUT BUFFER A

; PUSH CHARACTER ONTO RS232 OUTPUT BUFFER A

; IS CHANNEL = A?

; IS ACCUMULATOR = 1?

; TRANSFER KEYBOARD 2 ADDRESSES TO LOCAL

; TRANSFER KEYBOARD 1 ADDRESSES TO LOCAL

; SCAN FULL KEYBOARD

; NO KEY

; IS KEY = CTRL?

; IS KEY = SHIFT?

; IS KEY = SHIFT?

; ADD OFFSET TO KEY

; GET ASCII CODE FROM TABLE

; IS KEY VALID?

; IS KEY SAME AS DEBOUNCE CHARACTER

; BNE KEYBOARD
SUCCESSFUL GET KEY

DEBOUNCE NO

SCAN REST OF KEYBOARD

GOT, SCAN FULL KEYBOARD

KBD10 (OLDCHR), Y
CMF (OLDCHR), Y
BEQB (OLDCHR), Y
STA (ROWADR), Y
LDA (CBLADR), Y
EOR $00

NO KEY G0T

SCAN LDA #FE
JMP SCAN1
SCAN FULL KEYBOARD

SCAN LDA #FD
SCAN REST OF KEYBOARD

SCAN LDA #00

SCAN3 LDA ROW

SCAN4 INX

LSR SCAN4

BCS SCAN5

TAX

ASL TAA

ASL A

SEC A

RO TD

+---------------------------------------------------------------------+
| 459 KTABLE | HEX 20FFFF0D | ; SP X ER X ER X CR |
| 460 HEX FFFFFFFFFF | ; (ER X ER X ER X ER) |
| 461 HEX FFFFFF | ; (ER X ER X ER X ER) |
| 462 HEX FFFF08FF | ; (ER X ER X BS X ER) |
| 463 HEX 2E6C702D | ; (LXP) |
| 464 HEX 3A303B2F | ; 07 |
| 465 HEX 6D6A696F | ; (M X J X I X O) |
| 466 HEX 393B6B2C | ; 9B(K), |
| 467 HEX 62677975 | ; (B X G X Y X U) |
| 468 HEX 3736686E | ; 76(H X N) |
| 469 63 64 72 | |
+---------------------------------------------------------------------+
| FF99:  | 74 | 469 | HEX 63647274 | (C X D X R X T) |
| FF9A:  | 35 | 34 | 66 | 470 | HEX 33346676 | 54 F V |
| FF9B:  | 76 | 61 | 77 | 471 | HEX 7A617765 | (Z X A X W X E) |
| FF9C:  | 69 | 32 | 73 | 472 | HEX 33327378 | 32 (S X X) |
| FF9D:  | FF | FF | FF | 473 | HEX FFFF1871 | (ER X ER X ESC Q) |
| FF9E:  | FF | FF | FF | 474 | HEX 31F3F2F1 | 1 (F3 X F2 X F1) |

**CTRL**

| FF9F:  | FF | FF | FF | 475 | HEX 31F3F2F1 | 1 (F3 X F2 X F1) |

**UPPER CASE**

| FF9F:  | FF | FF | FF | 476 | HEX 31F3F2F1 | 1 (F3 X F2 X F1) |
```assembly
LDA 21H
JMP 22H

RTC

LUI KIHBW
OUT KIHBW
INC KIHBFI

BUFF IS EMPTY

RTC

LUI KIBIF
INC KIBRD

BUFF IS EMPTY

RTC

LUI KIBRD
INC KIHBFI

BUFF IS EMPTY

I2PFI LUI 12BUFI
BNE I2PFI

BUFF IS EMPTY

I2PFI LUI 12BUFI
BNE I2PFI

BUFF IS EMPTY

I2PFI LUI 12BUFI
INC 12BUFI

BUFF IS FULL

CPPUSH

CPPUSH

BUFF IS FULL

CPPUSH

BUFF IS FULL

CPPUSH

BUFF IS FULL
```
FROM RS232 INPUT BUFFER

- Buffer is full

- Character is full

- Character = buffer

- Push character onto input buffer

- Push onto display

- Backspace, carriage return

- No characters, test range

- Send character to input buffer

- Send character to input buffer

- Change channel

- Command delimiter

- Check if any characters

- No more characters

- Test range

- Carriage return?

- Push onto display

- Backspace

- Unrecognised command
FCB: A2 00 752 DISPL LDX #00
FCB: A0 00 753 LBY #00
FCF: A1 A7 754 LDA (CHAN,X) ; GET CURRENT OUTPUT DIRECTION (A,B)
FCF: C0 08 635 CMP #0B
FCF: A9 4B 757 BEQ DISPL1
FCF: 4C B0 FC 758 JMP DISPL2
FCA: A9 41 759 DISPLL LDX #1
FCA: 91 A3 760 DISPL2 STA (DISP),Y ; 16 CHARACTERS
FCA: 80 10 761 LBY #10
FCA: 88 1
FCI: 81 A3 763 LDA (DISP),Y URA #00 ; NO CURSOR
FCI: 81 A9 764 STA (DSPDAT,X)
FCI: 89 0F 765 TYA
FCI: 09 20 766 EOR #0F ; INVERT ADDRESS
FCC: 81 AB 767 URA #00 ; NO WRITE
FCC: 47 DF 768 AND #0F ; WRITE LOW
FCD: 81 A8 769 STA (DSPADR,X)
FCD: 09 20 770 ORA #20 ; WRITE LATCH
FCE: 81 AB 771 STA (DSPADR,X)
FCD: 6C 00 00 772 CPY #00
FCF: E0 0E 773 BNE DISPL3
FCF: 60 775 BNE DISPL3
FCF: 778 779 780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
FAC: A9 4C 835 SET1 LDA #1100
F12E: 85 A1 8332 STA #IP0P+1
F130: A9 FB 833 LDA #>1IP0P
F132: 85 A2 834 STA #IP0P+2
F134: A7 00 835 LDA #DSP1
F136: 85 A3 836 STA DISP
F138: 89 00 837 LDA #DSP1WT
F13A: 85 A5 838 STA DSPWT
F13C: A7 4A 839 LDA #CHAN1
F13E: 85 A7 840 STA CHAN
F140: A7 82 841 LDA #TSPACE
F142: 85 AF 842 STA TSPACE
F144: A9 80 843 LDA #CHAN
F146: 85 B1 844 STA CHN
F148: A7 01 845 LDA #DSP1DAT
F14A: 85 A9 846 STA DSP1DAT
F14C: A9 90 847 LDA #DSP1DAT
F14E: 85 AA 848 STA DSP1DAT+1
F150: A9 00 849 LDA #DSP1ADR
F152: 85 AB 850 STA DSPADR
F154: A7 90 851 LDA #DSP1ADR
F156: 85 AC 852 STA DSPADR+1
F158: A9 01 853 LDA #01
F15A: 85 AD 854 LDA #31
F15C: A9 31 855 STA TKNUM
F15E: 85 AE 856 LDA #31
F160: A7 2E 857 LDA #RIAPOP
F162: 85 B4 858 STA RIPOP+1
F164: A9 FB 859 LDA #RIAPOP
F166: 85 B5 860 STA RIPOP+2
F168: 60 861 RTS

***************
F16A: A9 5B 865 STA #12POF
F16B: 85 A1 866 LDA #12POF
F16D: A9 FB 867 LDA #12POF
F16F: 85 A2 868 STA #IP0P+2
F171: A9 20 869 LDA #DSP2
F173: 85 A3 870 LDA #DSP2
F175: A9 41 871 LDA #DSP2WT
F177: 85 A5 872 LDA #DSP2WT
F179: A7 48 873 LDA #CHAN2
F17B: 85 A7 874 LDA #CHAN2
F17D: A9 83 875 LDA #TSPACE
F17F: 85 AF 876 LDA #TSPACE
F181: A9 81 877 LDA #CHAN
F183: 85 B1 878 STA CHN
F185: A9 01 879 LDA #DSP2DAT
F187: 85 A9 880 STA DSPDAT
F189: A9 R0 881 LDA #DSP2DAT
F18B: 85 AA 882 LDA #DSP2DAT+1
F18D: A9 00 883 LDA #DSP2ADR
F18F: 85 AB 884 LDA #DSP2ADR
F191: A9 90 885 LDA #DSP2ADR
F193: 85 AC 886 LDA #DSP2ADR+1
F195: A9 02 887 LDA #02
F197: 85 AD 888 LDA #02
F199: A9 32 889 LDA #32
F19B: 85 AE 890 LDA #02
F19D: A9 30 891 LDA #02
F19F: 85 B4 892 LDA #02
F1A1: A9 FB 893 LDA #02
F1A3: 85 B5 894 LDA #02
F1A5: 60 895 RTS

***************
F1A6: A9 04 899 LDA #WAITTIM ; OPEN DOOR
F1A8: 78 900 SEL
F1A9: 85 91 901 STA WAITCNT
F1AA: A9 04 902 LDA #DOPENTIM
F1AB: 85 90 903 STA OPENTIM
F1AC: A9 64 904 LDA #64
F1AF: 85 99 905 STA HUND
F1B1: 58 906 CLI
F1B3: 40 907 RTS
F1B5: A0 00 908 LDOOR LDA SWORTB ; CHECK DOOR
F1B8: 29 90 909 AND #80
F1BA: 85 91 910 STA DORDEB
F1BB: F0 911 BEO DORCH1
TURN DOOR HAS CHARACTER TO BOTH UNIX CLRCUR "0"
; SEND WARNING COUNT

**CHECK WARNING**

**RESET WARNING FLAG**

**SEND 'W'**

**SEND CHARACTER TO BOTH UNIX AND RTS**

**CLEAR CURSOR**

**WRITE HIGH**

**CHECKSUM ON EPROM**

**SEND CHARACTER TO BOTH UNIX**

**ADD UP CHECKSUM NUMBER**

**SEND CHECKSUM**
DGOF: STA DOOR TIMER INTERRUPT
LDA WAITOPEN DOOR

HI SWPORTB

FO FLAG

LDA IF

INC ROGBRD

STA RSBTCH

LDA RSBUFY

INC ROGBRD

STA RSBTCH

RTI

FD02: 4C 0F FE
1100 JMP RSBTCH

1101 ****************************************
1102 # DOOR TIMER INTERRUPT *
1103 # HANDLER *
1104 ****************************************

FF05: AD 00 20
1105 TDINT LDA TDCL ; CLEAR INTERRUPT FLAG
1106 CLI ; ENABLE OTHER INTERRUPTS
1107 DEB RND
1108 DEB TDINT2
1109 PLA
1110 RTI

FF06: AD 00 20
1111 TDINT2 LDA #+$4 ; RESTART SECOND COUNTER
1112 STA HUND

FF07: AD 00 20
1113 TDINT3 LDA WARNCNT ; CHECK IF WARNING TIMEOUT
1114 BEQ TDINT4
1115 BIC WARNCNT
1116 BNE TDINT4

FF08: AD 00 20
1117 TDINT4 LDA IF
1118 STA WARNFLG ; SET WARNING FLAG
1119 PLA
1120 DEB TDINT5
1121 DEB WARNCNT
1122 BNE TDINT1 ; NOT FINISHED WAIT

FF09: AD 00 20
1123 PLA
1124 AND #$F
1125 STA SWPORTB ; OPEN DOOR
1126 PLA
1127 RTI

FF0A: AD 00 20
1128 TDINT5 LDA OPENCNT ; CHECK IF HAVE TO CLOSE DOOR
1129 BEQ TDINT6
1130 BIC OPENCNT
1131 BNE TDINT1

FF0B: AD 00 20
1132 TDINT6 LDA SWPORTB ; CLOSE DOOR
1133 ORA #$40
1134 STA SWPORTB
1135 PLA
1136 RTI

1137 ****************************************
1138 # LIGHTS OFF *
1139 ****************************************

FF43: AD 00 20
1140 LOFF LDA SWPORTB

FF44: AD 00 20
1141 AND #$F
1142 STA SWPORTB

FF45: AD 00 20
1143 RTI

1144 ****************************************
1145 # MAIN INTERRUPT HANDLER *
1146 ****************************************

FF47: 4B
Symbol table - alphabetical order:

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Address</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACRB</td>
<td>$800B</td>
<td>Break</td>
</tr>
<tr>
<td>CHAN1</td>
<td>$81</td>
<td>Channel</td>
</tr>
<tr>
<td>CHKSUM</td>
<td>$FEB4</td>
<td>Checksum</td>
</tr>
<tr>
<td>COLADR</td>
<td>$4C</td>
<td>Column</td>
</tr>
<tr>
<td>COM</td>
<td>$8F7</td>
<td>Command</td>
</tr>
<tr>
<td>COM3</td>
<td>$FC19</td>
<td>Command3</td>
</tr>
<tr>
<td>COM4</td>
<td>$FC48</td>
<td>Command4</td>
</tr>
<tr>
<td>CHR21</td>
<td>$FC99</td>
<td>Character</td>
</tr>
<tr>
<td>DFL</td>
<td>$43</td>
<td>Display</td>
</tr>
<tr>
<td>DISP3</td>
<td>$20</td>
<td>Display3</td>
</tr>
<tr>
<td>DOORCH1</td>
<td>$F1C</td>
<td>Doorch1</td>
</tr>
<tr>
<td>DPS</td>
<td>$F5</td>
<td>Door</td>
</tr>
<tr>
<td>DSPDRA</td>
<td>$003</td>
<td>DSPDRA</td>
</tr>
<tr>
<td>DSPDAT</td>
<td>$8001</td>
<td>DSPDAT</td>
</tr>
<tr>
<td>DSPOFF</td>
<td>$A8</td>
<td>DSPOFF</td>
</tr>
<tr>
<td>HTA</td>
<td>$01</td>
<td>HTA</td>
</tr>
<tr>
<td>ILBUF</td>
<td>$6600</td>
<td>ILBUF</td>
</tr>
<tr>
<td>IIPSH</td>
<td>$F86C</td>
<td>IPSP</td>
</tr>
<tr>
<td>I/SUB</td>
<td>$70</td>
<td>I/SUB</td>
</tr>
<tr>
<td>I/PFSH1</td>
<td>$FA6</td>
<td>IPFSH1</td>
</tr>
<tr>
<td>I/SUB1</td>
<td>$001</td>
<td>I/SUB1</td>
</tr>
<tr>
<td>INIT1</td>
<td>$F23</td>
<td>INIT1</td>
</tr>
<tr>
<td>INIT5</td>
<td>$32D</td>
<td>INIT5</td>
</tr>
<tr>
<td>KBD0</td>
<td>$F9CE</td>
<td>KBD0</td>
</tr>
<tr>
<td>KBD1</td>
<td>$F9CE</td>
<td>KBD1</td>
</tr>
<tr>
<td>KBD2</td>
<td>$F9CE</td>
<td>KBD2</td>
</tr>
<tr>
<td>KBD3</td>
<td>$F9CE</td>
<td>KBD3</td>
</tr>
<tr>
<td>KBD4</td>
<td>$F9CE</td>
<td>KBD4</td>
</tr>
<tr>
<td>KBD5</td>
<td>$F9CE</td>
<td>KBD5</td>
</tr>
<tr>
<td>KBD6</td>
<td>$F9CE</td>
<td>KBD6</td>
</tr>
<tr>
<td>LINT2</td>
<td>$F7F</td>
<td>LINT2</td>
</tr>
<tr>
<td>LINT4</td>
<td>$FBB8</td>
<td>LINT4</td>
</tr>
<tr>
<td>MINT4</td>
<td>$FBB8</td>
<td>MINT4</td>
</tr>
<tr>
<td>MINT6</td>
<td>$FBB8</td>
<td>MINT6</td>
</tr>
<tr>
<td>MINT7</td>
<td>$3F99</td>
<td>MINT7</td>
</tr>
<tr>
<td>MINT8</td>
<td>$3F99</td>
<td>MINT8</td>
</tr>
<tr>
<td>MINT9</td>
<td>$3F99</td>
<td>MINT9</td>
</tr>
<tr>
<td>MINT10</td>
<td>$3F99</td>
<td>MINT10</td>
</tr>
<tr>
<td>MDOR</td>
<td>$F65</td>
<td>MDOR</td>
</tr>
<tr>
<td>OLDCRH2</td>
<td>$5B</td>
<td>OLDCRH2</td>
</tr>
<tr>
<td>RSEG</td>
<td>$FFFC</td>
<td>RSEG</td>
</tr>
<tr>
<td>RBUF1</td>
<td>$F01</td>
<td>RBUF1</td>
</tr>
<tr>
<td>RBUF2</td>
<td>$F02</td>
<td>RBUF2</td>
</tr>
<tr>
<td>RBUF3</td>
<td>$F03</td>
<td>RBUF3</td>
</tr>
<tr>
<td>RBUF4</td>
<td>$F04</td>
<td>RBUF4</td>
</tr>
<tr>
<td>RBUF5</td>
<td>$F05</td>
<td>RBUF5</td>
</tr>
<tr>
<td>RSACT1</td>
<td>$E000</td>
<td>RSACT1</td>
</tr>
<tr>
<td>S/SEC</td>
<td>$8000</td>
<td>S/SEC</td>
</tr>
<tr>
<td>S/AIR</td>
<td>$0000</td>
<td>S/AIR</td>
</tr>
<tr>
<td>S/CHAP</td>
<td>$003</td>
<td>S/CHAP</td>
</tr>
<tr>
<td>SC/21</td>
<td>$000</td>
<td>SC/21</td>
</tr>
<tr>
<td>SCAN2</td>
<td>$FA2</td>
<td>SCAN2</td>
</tr>
<tr>
<td>SEND</td>
<td>$FC9C</td>
<td>SEND</td>
</tr>
<tr>
<td>START1</td>
<td>$20</td>
<td>START1</td>
</tr>
<tr>
<td>SWCH</td>
<td>$C000</td>
<td>SWCH</td>
</tr>
<tr>
<td>TECH</td>
<td>$001</td>
<td>TECH</td>
</tr>
<tr>
<td>TECH1</td>
<td>$009</td>
<td>TECH1</td>
</tr>
<tr>
<td>TECH2</td>
<td>$005</td>
<td>TECH2</td>
</tr>
<tr>
<td>TDINT1</td>
<td>$FF0F</td>
<td>TDINT1</td>
</tr>
<tr>
<td>TDINT6</td>
<td>$F39</td>
<td>TDINT6</td>
</tr>
<tr>
<td>TRCNT</td>
<td>$20</td>
<td>TRCNT</td>
</tr>
<tr>
<td>TRINT</td>
<td>$F0E5</td>
<td>TRINT</td>
</tr>
<tr>
<td>WARNCH</td>
<td>$92</td>
<td>WARNCH</td>
</tr>
<tr>
<td>WAPFLG</td>
<td>$98</td>
<td>WAPFLG</td>
</tr>
</tbody>
</table>

Symbol table - numerical order:

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Address</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISP2</td>
<td>$00</td>
<td>Disp2</td>
</tr>
<tr>
<td>IMPUT</td>
<td>$04</td>
<td>Input</td>
</tr>
<tr>
<td>MINT</td>
<td>$04</td>
<td>MINT</td>
</tr>
<tr>
<td>DSP2WT</td>
<td>$04</td>
<td>DSP2WT</td>
</tr>
<tr>
<td>DSP2</td>
<td>$20</td>
<td>DSP2</td>
</tr>
<tr>
<td>WAPFLG</td>
<td>$3C</td>
<td>WAPFLG</td>
</tr>
</tbody>
</table>

Errors: 0
APPENDIX E

PARTS LIST (+ SUGGESTED SUPPLIERS)

October 15, 1984
### MAIN BOARD

#### SEMICONDUCTORS

<table>
<thead>
<tr>
<th>Part No</th>
<th>Specification</th>
<th>Value</th>
<th>Supplier</th>
</tr>
</thead>
<tbody>
<tr>
<td>IC1</td>
<td>6502 CPU</td>
<td>5.67</td>
<td>ENERGY CONTROL</td>
</tr>
<tr>
<td>IC2</td>
<td>6522 VIA</td>
<td>5.34</td>
<td>ENERGY CONTROL</td>
</tr>
<tr>
<td>IC3</td>
<td>6522 VIA</td>
<td>5.34</td>
<td>ENERGY CONTROL</td>
</tr>
<tr>
<td>IC4</td>
<td>6522 VIA</td>
<td>5.34</td>
<td>ENERGY CONTROL</td>
</tr>
<tr>
<td>IC5</td>
<td>6522 VIA</td>
<td>5.34</td>
<td>ENERGY CONTROL</td>
</tr>
<tr>
<td>IC6</td>
<td>6522 VIA</td>
<td>5.34</td>
<td>ENERGY CONTROL</td>
</tr>
<tr>
<td>IC7</td>
<td>6551 ACIA</td>
<td>6.03</td>
<td>ENERGY CONTROL</td>
</tr>
<tr>
<td>IC8</td>
<td>6551 ACIA</td>
<td>6.03</td>
<td>ENERGY CONTROL</td>
</tr>
<tr>
<td>IC9</td>
<td>2732 EPROM</td>
<td>6.05</td>
<td>ENERGY CONTROL</td>
</tr>
<tr>
<td>IC10</td>
<td>6116 RAM</td>
<td>8.68</td>
<td>ENERGY CONTROL</td>
</tr>
<tr>
<td>IC11</td>
<td>1488 RS232TX</td>
<td>2.00</td>
<td>VINCON</td>
</tr>
<tr>
<td>IC12</td>
<td>1489 RS232RX</td>
<td>2.00</td>
<td>VINCON</td>
</tr>
<tr>
<td>IC13</td>
<td>4049 HEXINU</td>
<td>0.80</td>
<td>DICK SMITH</td>
</tr>
<tr>
<td>IC14</td>
<td>4013 DFlipFlop</td>
<td>0.80</td>
<td>DICK SMITH</td>
</tr>
<tr>
<td>IC15</td>
<td>74LS04 HEXINU</td>
<td>1.50</td>
<td>DICK SMITH</td>
</tr>
<tr>
<td>IC16</td>
<td>74LS138 DECODER</td>
<td>1.15</td>
<td>VINCON</td>
</tr>
<tr>
<td>IC17</td>
<td>UA73005 5V5ADEC</td>
<td>7.75</td>
<td>DICK SMITH</td>
</tr>
<tr>
<td>IC18</td>
<td>74LS22 QUADOR</td>
<td>1.50</td>
<td>DICK SMITH</td>
</tr>
<tr>
<td>IC19</td>
<td>74LS138 DECODER</td>
<td>1.15</td>
<td>DICK SMITH</td>
</tr>
<tr>
<td>D1</td>
<td>IN5404 RECT</td>
<td>0.45</td>
<td>DICK SMITH</td>
</tr>
<tr>
<td>D2</td>
<td>IN5404 RECT</td>
<td>0.45</td>
<td>DICK SMITH</td>
</tr>
<tr>
<td>D3</td>
<td>IN5404 RECT</td>
<td>0.45</td>
<td>DICK SMITH</td>
</tr>
<tr>
<td>D4</td>
<td>IN5404 RECT</td>
<td>0.45</td>
<td>DICK SMITH</td>
</tr>
<tr>
<td>D5</td>
<td>IN5404 RECT</td>
<td>0.45</td>
<td>DICK SMITH</td>
</tr>
<tr>
<td>D6</td>
<td>IN5404 RECT</td>
<td>0.45</td>
<td>DICK SMITH</td>
</tr>
<tr>
<td>D8</td>
<td>BZX12 12VZENER</td>
<td>2.50</td>
<td>VINCON</td>
</tr>
<tr>
<td>D9</td>
<td>BZX12 12VZENER</td>
<td>2.50</td>
<td>VINCON</td>
</tr>
<tr>
<td>D10</td>
<td>IN914</td>
<td>1.15</td>
<td>DICK SMITH</td>
</tr>
<tr>
<td>D11</td>
<td>IN914</td>
<td>1.15</td>
<td>DICK SMITH</td>
</tr>
<tr>
<td>Q1</td>
<td>DS558 SIG</td>
<td>1.15</td>
<td>DICK SMITH</td>
</tr>
<tr>
<td>Q2</td>
<td>BD681 DARL</td>
<td>1.70</td>
<td>DICK SMITH</td>
</tr>
<tr>
<td>XT1</td>
<td>2MHz XTAL</td>
<td>2.86</td>
<td>ENERGY CONTROL</td>
</tr>
<tr>
<td>XT2</td>
<td>1.8432MHz XTAL</td>
<td>2.86</td>
<td>ENERGY CONTROL</td>
</tr>
<tr>
<td>XT3</td>
<td>1.8432MHz XTAL</td>
<td>2.86</td>
<td>ENERGY CONTROL</td>
</tr>
</tbody>
</table>

#### RESISTORS

<table>
<thead>
<tr>
<th>Resistor</th>
<th>Value</th>
<th>Supplier</th>
</tr>
</thead>
<tbody>
<tr>
<td>R2</td>
<td>2.2K 0.25W</td>
<td>0.05</td>
</tr>
<tr>
<td>R3</td>
<td>10K 0.25W</td>
<td>0.05</td>
</tr>
<tr>
<td>R4</td>
<td>4.7K 0.25W</td>
<td>0.05</td>
</tr>
<tr>
<td>R5</td>
<td>68 5W</td>
<td>0.50</td>
</tr>
<tr>
<td>R6</td>
<td>68 5W</td>
<td>0.50</td>
</tr>
<tr>
<td>R7</td>
<td>56 0.5W</td>
<td>0.06</td>
</tr>
<tr>
<td>R8</td>
<td>47K 0.25W</td>
<td>0.05</td>
</tr>
<tr>
<td>R9</td>
<td>100 0.25W</td>
<td>0.05</td>
</tr>
</tbody>
</table>

#### CAPACITORS

<table>
<thead>
<tr>
<th>Capacitor</th>
<th>Value</th>
<th>Supplier</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>0.1uF CERAMIC</td>
<td>0.18</td>
</tr>
<tr>
<td>C2</td>
<td>1000uF 25VCELECTRO</td>
<td>0.80</td>
</tr>
<tr>
<td>C3</td>
<td>1000uF 25VUTANT</td>
<td>1.70</td>
</tr>
<tr>
<td>C4</td>
<td>1000uF 40VCELECTRO</td>
<td>7.95</td>
</tr>
<tr>
<td>C5</td>
<td>10uF 25VUTANT</td>
<td>0.70</td>
</tr>
<tr>
<td>C6</td>
<td>10uF 25VUTANT</td>
<td>0.70</td>
</tr>
<tr>
<td>C7</td>
<td>1000uF 25VCELECTRO</td>
<td>0.80</td>
</tr>
<tr>
<td>C8</td>
<td>10uF 25VUTANT</td>
<td>0.70</td>
</tr>
<tr>
<td>C9</td>
<td>10uF 25VUTANT</td>
<td>0.70</td>
</tr>
<tr>
<td>C8-C20</td>
<td>0.1uF CERAMIC</td>
<td>2.34</td>
</tr>
</tbody>
</table>
* MISC

T1 PL24/40VA 2*12V-TRANS 17.50 VIMCO
F1 S-4206 FUSE-HELDER 1.95 DICK SMITH
G1 L-6210 REED-SWITCH 2.70 DICK SMITH
RS1 RS232 SOCKET 4.25 DICK SMITH
RS2 RS232 SOCKET 4.75 DICK SMITH

* SOCKETS

4* 14 PIN-DIL 1.40 DICK SMITH
5* 16 PIN-DIL 2.00 DICK SMITH
2* 24 PIN-DIL 1.10 DICK SMITH
2* 26 PIN-DIL 1.30 DICK SMITH
6* 40 PIN-DIL 4.30 DICK SMITH

* DISPLAY UNIT (LED DISPLAY TYPE)

* SEMICONDUCTORS

IC1 A69-1416 DISPLAY 17.47 ENERGY CONTROL
IC2 A69-1416 DISPLAY 17.47 ENERGY CONTROL
IC3 A69-1416 DISPLAY 17.47 ENERGY CONTROL
IC4 74LS138 DECODER 1.15 DICK SMITH

* CAPACITORS

C1 0.1uf CERAMIC 0.18 DICK SMITH
C2 0.1uf CERAMIC 0.18 DICK SMITH
C3 0.1uf CERAMIC 0.18 DICK SMITH
C4 0.1uf CERAMIC 0.18 DICK SMITH
C5 0.1uf CERAMIC 0.18 DICK SMITH

* SOCKETS

1* 16 PIN-DIL 0.40 DICK SMITH

* PLUGS

2* 16 PIN-DIL 5.90 DICK SMITH

* DISPLAY UNIT (LCD TYPE)

D1 LM-16251 LCD-DISPLAY 62.00 DAHEVA

* PLUGS

1* 16 PIN-DIL 2.95 DICK SMITH

* DOOR LOCK

* 12 VOLT, POWER OFF TO LOCK
* ELECTRIC DOOR STRIKE
* MODEL 14KL
* PART No. BS12080 35.00 WORMALD SECURITY

* ADDRESSES

** DICK SMITH DICK SMITH ELECTRONICS
** EXPRESS ORDER CENTRE
** PO BOX 321, NORTH RYDE, N.S.W.
** AUSTRALIA, 2113

** WORMALD SECURITY STEVE ALDERMAN
** WORMALD SECURITY
** 1 BELMORE ST JOLLONGONG 2500
** Ph 297111