

Utah State University

DigitalCommons@USU

Undergraduate Honors Capstone Projects

Honors Program

5-1994

The Design of an Augmentative Communications Device

Darren Blaser

Utah State University

Scott Sorenson

Utah State University

Follow this and additional works at: <https://digitalcommons.usu.edu/honors>



Part of the [Electrical and Computer Engineering Commons](#)

Recommended Citation

Blaser, Darren and Sorenson, Scott, "The Design of an Augmentative Communications Device" (1994).

Undergraduate Honors Capstone Projects. 320.

<https://digitalcommons.usu.edu/honors/320>

This Thesis is brought to you for free and open access by the Honors Program at DigitalCommons@USU. It has been accepted for inclusion in Undergraduate Honors Capstone Projects by an authorized administrator of DigitalCommons@USU. For more information, please contact digitalcommons@usu.edu.



**THE DESIGN OF AN AUGMENTATIVE
COMMUNICATIONS DEVICE**

by

Darren Blaser and Scott Sorenson

Thesis submitted in partial fulfillment
of the requirements for the degree

of

DEPARTMENT HONORS

in

Electrical Engineering

Utah State University

Logan, Utah

1994

Introduction

There is an extremely large number of people in the modern world who have difficulty communicating. Among these are approximately 25 million people with cerebral palsy. Cerebral palsy is a disability that is usually caused during, or shortly after birth. Cerebral palsy usually affects a person's motor skills and sense of balance, often rendering them incapable of speaking clearly enough to be understood well. Because this group of people cannot communicate as effectively as other individuals, many uninformed people suppose they do not have as great a need to speak, to say things like, "How are you?", "I'm Hungry", "Thank you" or "I love you!" All people, including those suffering from speaking impairments, have feelings and emotions that need to be expressed. Their desires to communicate are just as strong, if not stronger, than individuals who have greater freedom to express themselves. To assure the highest possible quality of life-style for both individuals with severe speech impediments and all those associated with them, means must be devised to augment the ability of these individuals to communicate. These means must not only be easy and convenient to use, they must also be affordable. With these requirements in mind, we designed an augmentative communications device which we call Talk4Me. Talk4Me allows a user to playback prerecorded messages, enabling them to concisely express a limited number of ideas.

Table of Contents

Table of Contents	i
List of Figures	ii
Executive Summary	iii
Introduction	1
Project Implementation	2
The User Interface	2
The Audio Subsystem	4
System Controller	7
The Power Supply	9
Results and Recommendations	12
Appendix	14

List of Figures and Tables

Figure 1. System Block Diagram	2
Figure 2. Control Panel	3
Figure 3. Membrane Switch Panel with LEDs	4
Figure 4. Audio Subsystem	5
Figure 5. ISD1020A Simplified Block Diagram	6
Figure 6. System Controller	7
Figure 7. Software Modules	8
Figure 8. Rechargeable Power Supply	10
Table 1. Current Draw and Power Consumption	9

Executive Summary

Systems for augmenting the communicative abilities of individuals with severe speech impediments have been developed by several companies. However, most of these systems cost more than \$1000, which prohibits many individuals from taking advantage of them. In October of 1992, Kathy Thurston proposed the investigation of producing a portable communication aid for the use of her sister for under \$1000. The purpose of this report is to present the design of this augmentative communications device, which is called Talk4Me.

Although Talk4Me was designed specifically for Kathy's sister, whose severe speech impairment was caused by cerebral palsy, it has been found that Talk4Me could be of great use by many other individuals with speech impairments from a wide variety of causes.

The developed system consists of a user interface, designed to be easy and convenient for the user, and a record/playback system which records various phrases and plays them back when selected. It also includes a rechargeable battery system for portability. The results of this investigation and design indicate that an augmentative communications device is achievable for under \$1000. However, some modifications to the design are required for Talk4Me to be manufactured.

Introduction

There is an extremely large number of people in the modern world who have difficulty communicating. Among these are approximately 25 million people with cerebral palsy. Cerebral palsy is a disability that is usually caused during, or shortly after birth. Cerebral palsy usually affects a person's motor skills and sense of balance, often rendering them incapable of speaking clearly enough to be understood well. Because this group of people cannot communicate as effectively as other individuals, many uninformed people suppose they do not have as great a need to speak, to say things like, "How are you?", "I'm Hungry", "Thank you" or "I love you!" All people, including those suffering from speaking impairments, have feelings and emotions that need to be expressed. Their desires to communicate are just as strong, if not stronger, than individuals who have greater freedom to express themselves. To assure the highest possible quality of life-style for both individuals with severe speech impediments and all those associated with them, means must be devised to augment the ability of these individuals to communicate. These means must not only be easy and convenient to use, they must also be affordable. With these requirements in mind, we designed an augmentative communications device which we call Talk4Me. Talk4Me allows a user to playback prerecorded messages, enabling them to concisely express a limited number of ideas.

Project Implementation

The preceding section discusses the need to develop a portable communication aid that is easy and convenient to use. The following report presents the Talk4Me augmentative communication device, which was developed for this application, as four functional subsystems: 1) the user interface, 2) the audio subsystem, 3) the system controller, and 4) the power supply. Each of these subsystems is shown in the system block diagram, as shown in Figure 1.

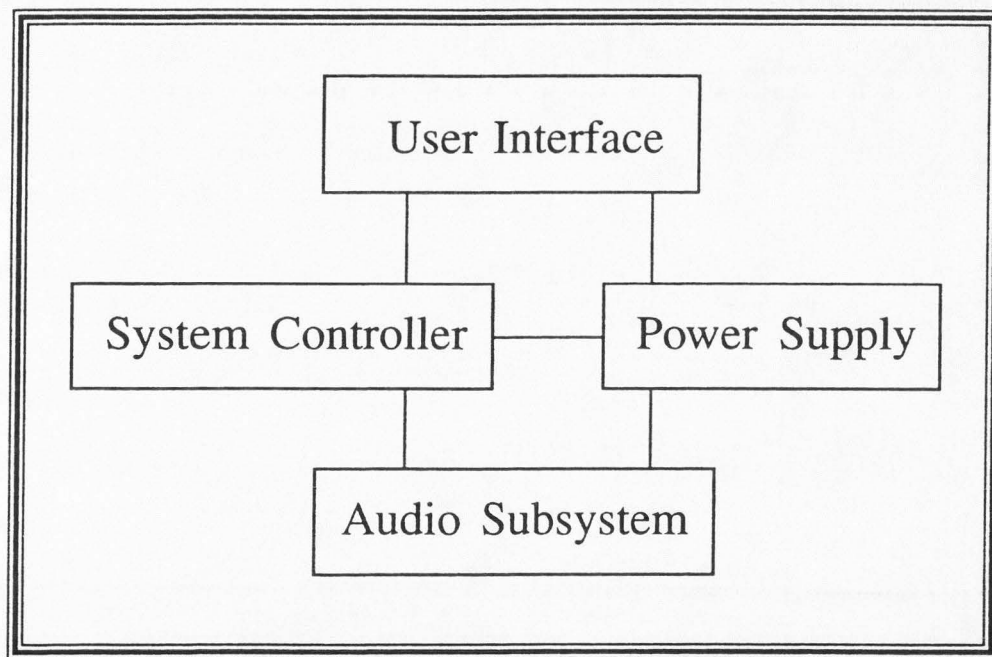


Figure 1. System Block Diagram

The User Interface

Ease of use is a major factor for the success of any communication device. Therefore, the user interface must be simple and easy to use. Talk4Me has a user interface that consists of a membrane switch panel and a control panel. To use Talk4Me, the user first turns on the unit with the push button on/off switch located

on the front panel near the membrane switches. The user then checks the switches on the control panel to be sure the correct options are selected. Figure 2 shows the six toggle switches which are available to select the various options. The available

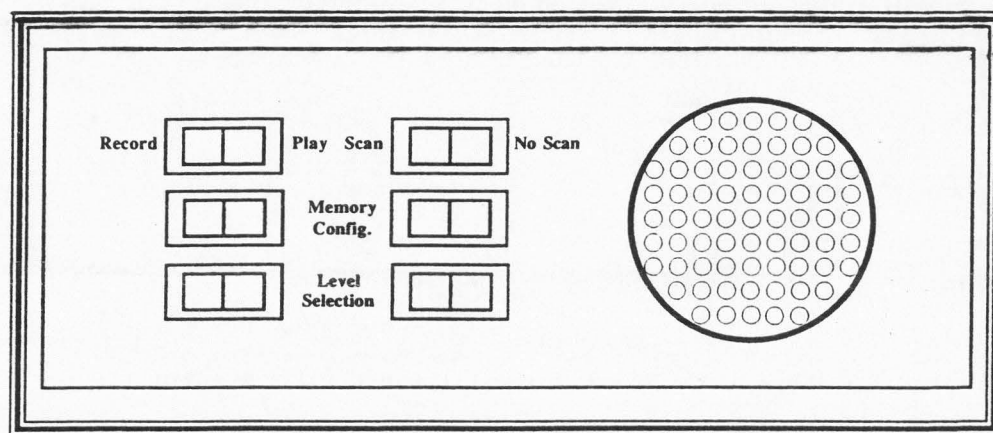


Figure 2. Control Panel

options include: three possible modes of operation (selected by the top two switches), four possible memory configurations (selected by the middle two switches), and a level selection of four possible levels for quad-level operation with toggle switch level selection (selected by the two switches at the bottom of the control panel). If the user wishes to record a message, she simply presses the play/record rocker switch into the record position. This selects the recording mode. She then presses and holds the appropriate membrane switch down, while repeating the desired message. The message is recorded as long as the membrane switch is firmly depressed or until the allotted time for that switch is depleted. To play the message back she presses the play/record switch into the play position and presses the appropriate membrane switch. In the scanning mode, Talk4Me uses a simple row-column keyboard scan which allows the user to select the desired message. Each row lights sequentially for two seconds. This continues until any switch is pressed. After a switch is depressed, the LED in each column sequentially lights for two seconds, until another switch is pressed. When any switch is pressed for the second time,

Talk4Me plays back the message corresponding to the switch by the LED that is lighted.

Figure 3 shows the membrane switch panel. The small square boxes shown

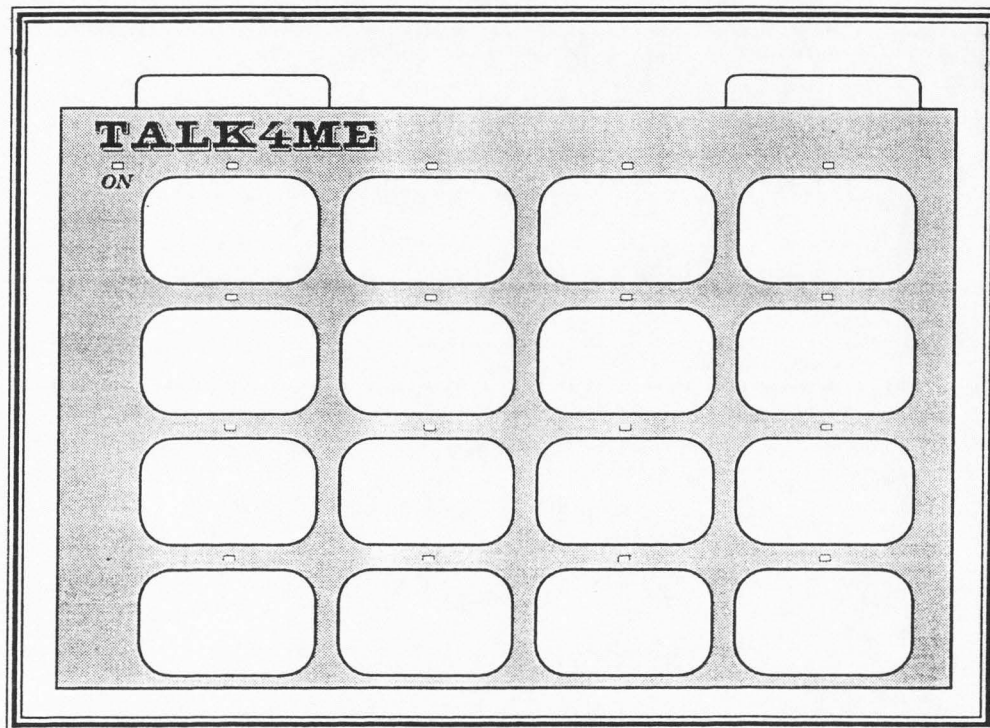


Figure 3. Membrane Switch Panel with LEDs

are LEDs which light while the device is recording and while the device is operating in the scanning mode, providing ease of operation for the user. The larger rectangles with rounded corners represent the actual membrane switch locations. There is also a LED in the top left hand corner of Figure 3 which functions as an on/off light.

The Audio Subsystem

The audio subsystem consists of an electret microphone, preamplifier, 16 ohm speaker and an array of ISD (Information Storage Devices Inc.) storage chips.

A complete schematic of the audio subsystem is shown in Figure 4. The electret microphone, with a sensitivity of 64 volts per millibar, captures the audio signal from the user and converts it into an electrical signal. This signal is then boosted

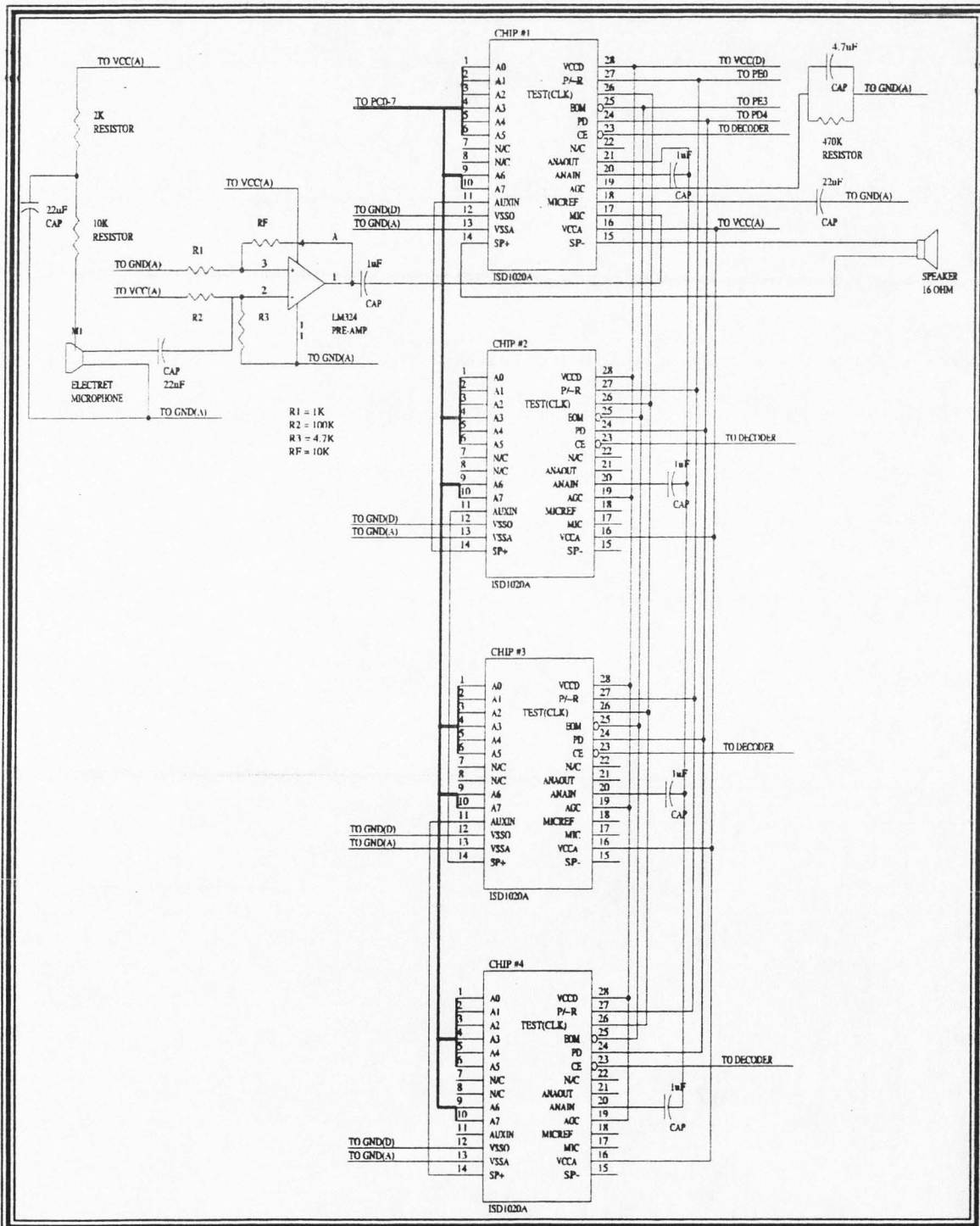


Figure 4. Audio Subsystem

by a low power preamplifier with a voltage gain of 16. This preamplification is accomplished with an LM324 operational amplifier in a non-inverting configuration which is biased for a 2.25 volt nominal output voltage with no input. The output of the preamplifier is capacitively coupled to the ISD storage array with a one microfarad capacitor. The ISD storage array consists of four ISD1020A single-chip voice record/playback devices. This is a new technology which utilizes patented Direct Analog Storage Technology (DAST). This revolutionary EEPROM storage method allows analog data to be written directly into a single cell without analog-to-digital or digital-to-analog conversion. This results in both an increase in density over equivalent digital methods and non-volatile storage of analog data. Figure 5 shows the simplified block diagram of the ISD1020A. The audio input comes into the

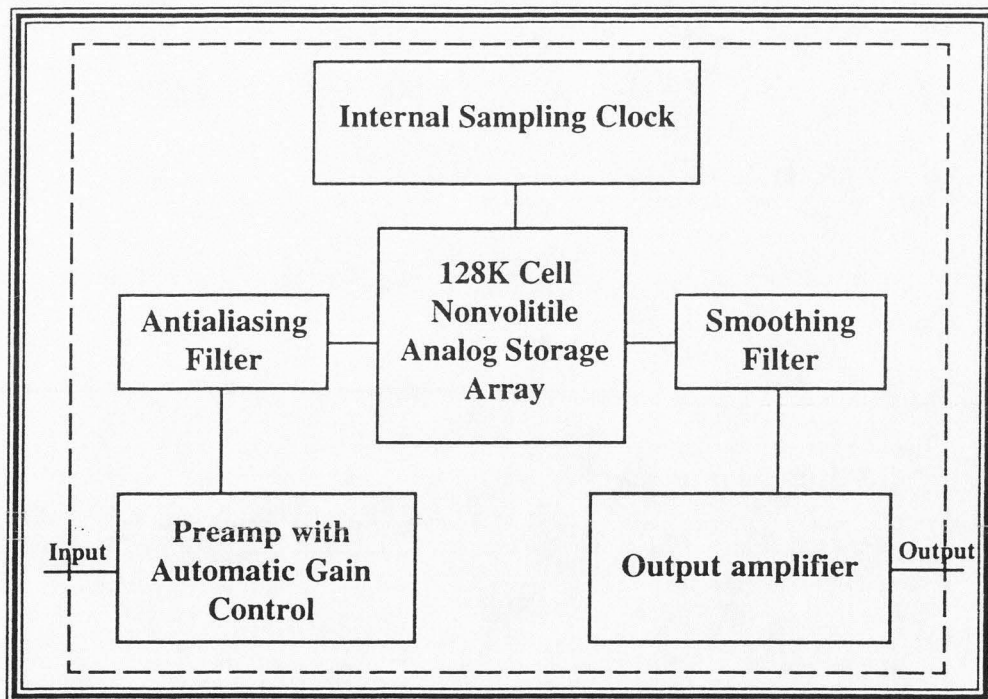


Figure 5. ISD1020A Simplified Block Diagram

preapmlifier whose gain is automatically adjusted to maintain an optimum signal level into the filter. The anti-aliasing filter is a 5 pole low-pass filter with a roll-off of 40 dB per octave above the cut-off frequency of 3.4 kHz. The internal clock sets

the audio sampling rate at 6.4 kHz. Thus the anti-aliasing filter removes input frequency components above half the sampling frequency, satisfying the nyquist criterion. The audio signal is then written into the nonvolatile analog storage array. During playback, the recorded analog voltages are sequentially read from the storage array. The smoothing filter removes the sampling frequency component and the original audio signal is then passed into the output power amplifier which provides about 50 milliwatts RMS into two eight ohm speakers. An additional power amplifier is not needed as the speakers we are using have 93 dB per watt sensitivity at .5 meters. Because the ISD storage chips employ electrically erasable cmos technology the memory is nonvolatile, so even if the system is completely powered down there is no resulting loss of memory.

The System Controller

The heart of the system controller is the motorolla HC811E2 microcontroller. Refer to Figure 6.

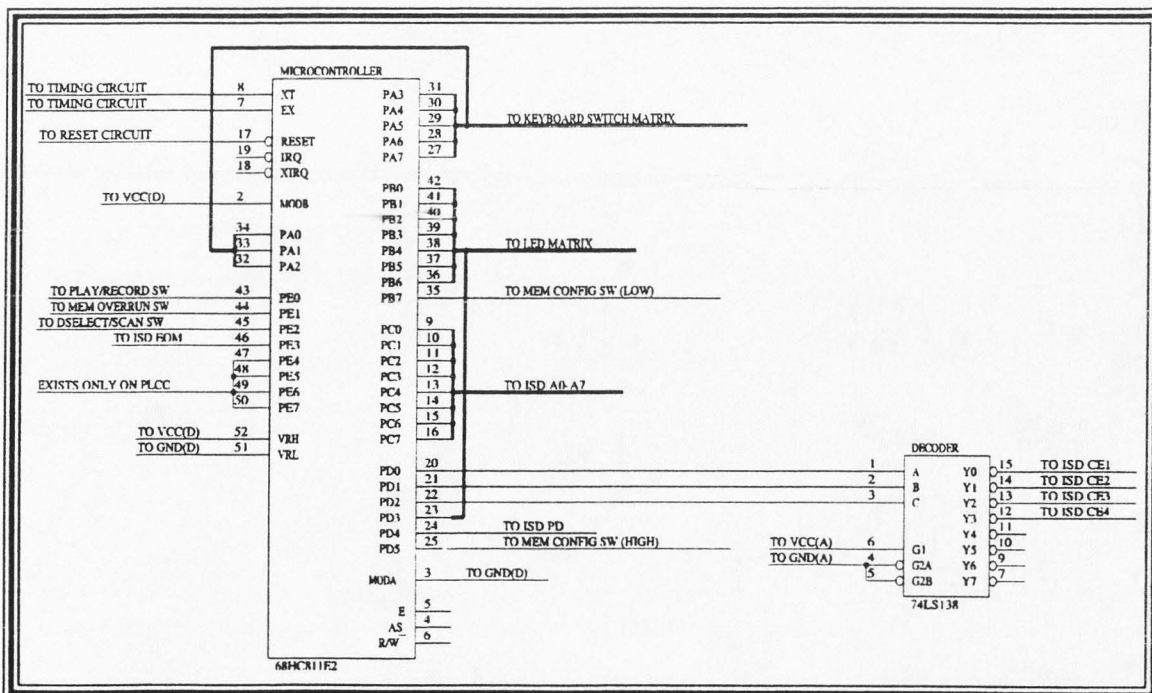


Figure 6. System Controller

The motorolla HC811E2 microcontroller has five i/o ports which consists of 16 output and 17 input lines. Port a (pins 0-6) and port d (pin 3) are connected to the membrane switch array. These lines are used to scan the membrane switch panel. Port b is used strictly for output and drives the bank of LEDs on the front panel. Port c is used as the address bus and is tied directly to the ISD storage array. Ports d and e are used to scan the control panel and send the required control signals to the ISD storage array.

The motorolla HC811E2 microcontroller has a full two kilobytes of electrically erasable memory available for programming. A special circuit was built to program the microcontroller from any IBM pc-compatible computer with a serial port. The program for the microcontroller was written in motorolla assembly code, then compiled and transferred serially to the interfacing circuitry, which programs the microcontroller. A complete listing of the code used for Talk4Me is provided in the appendix.

The software for Talk4Me was written in modular fashion with three major modules, as shown in Figure 7. The initialization module initializes each of the ports, the stack, and various variables that were used in the static ram memory.

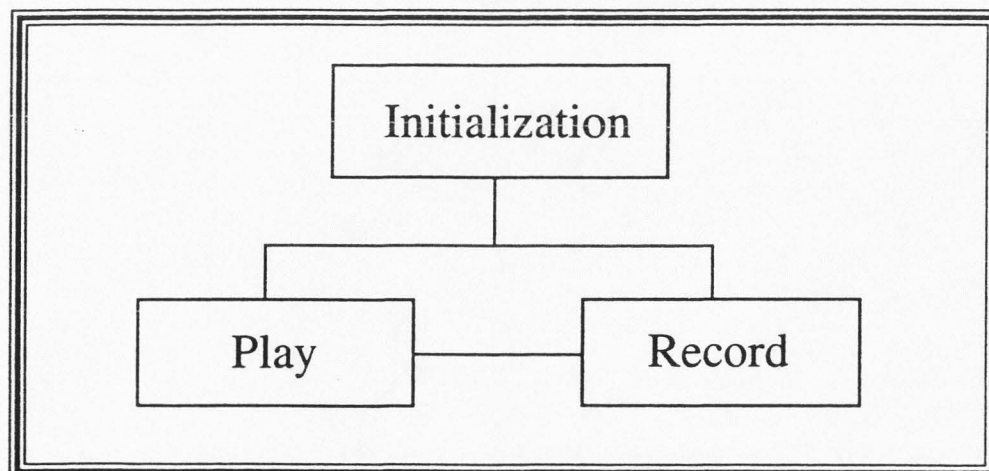


Figure 7. Software Modules

This module also reads the configuration switches in order to begin operation in the proper operating mode. After completing the initialization routine, the program continues with either the play or the record module as selected. Both the play and the record modules call subroutines which scan the membrane switch panel, calculate and place appropriate addresses on the address bus for the ISD storage array, enable the appropriate ISD storage chip, and wait for given delay times necessary for the control signals required for proper operation of the ISD storage array.

When the microcontroller was received it was found that the dip package had four less input pins than the PLCC package that was detailed in the data books. We corrected this problem by using a three to eight decoder to provide the chip enable signals for the ISD storage array, as shown previously in Figure 6.

The Power Supply

Talk4Me is a portable battery operated unit which uses five rechargeable nickel cadmium batteries. Table 1 shows the current draw and the power consumption measured during five different loading conditions. It is estimated that Talk4Me will be in the Direct Select (wait state) between 90 and 95 percent of the time. This gives Talk4Me an approximate lifetime of 7 days continuous use on a single charge. This

Mode of Operation	Current Drawn (mA)	Power Consumption (mW)
Direct Select (wait state)	18.8	113
Direct Select (silent play)	100	600
Direct Select (loud play)	140	840
Record (wait state)	23.9	143
Record (active)	87.0	522

Table 1. Current Draw and Power Consumption

extended lifetime adds to the convenience of the user, as recharging would only be required weekly with heavy use.

Figure 8 shows the schematic for the batteries and voltage regulator used in the power supply. All the chips on the circuit board require a single +5 volt supply. This is delivered to the circuit board by the LM2931AT-5.0 low drop off voltage regulator with a variation of only ± 0.25 volts. The input voltage for this regulator comes directly from the five batteries as shown in Figure 8. Each nickel cadmium

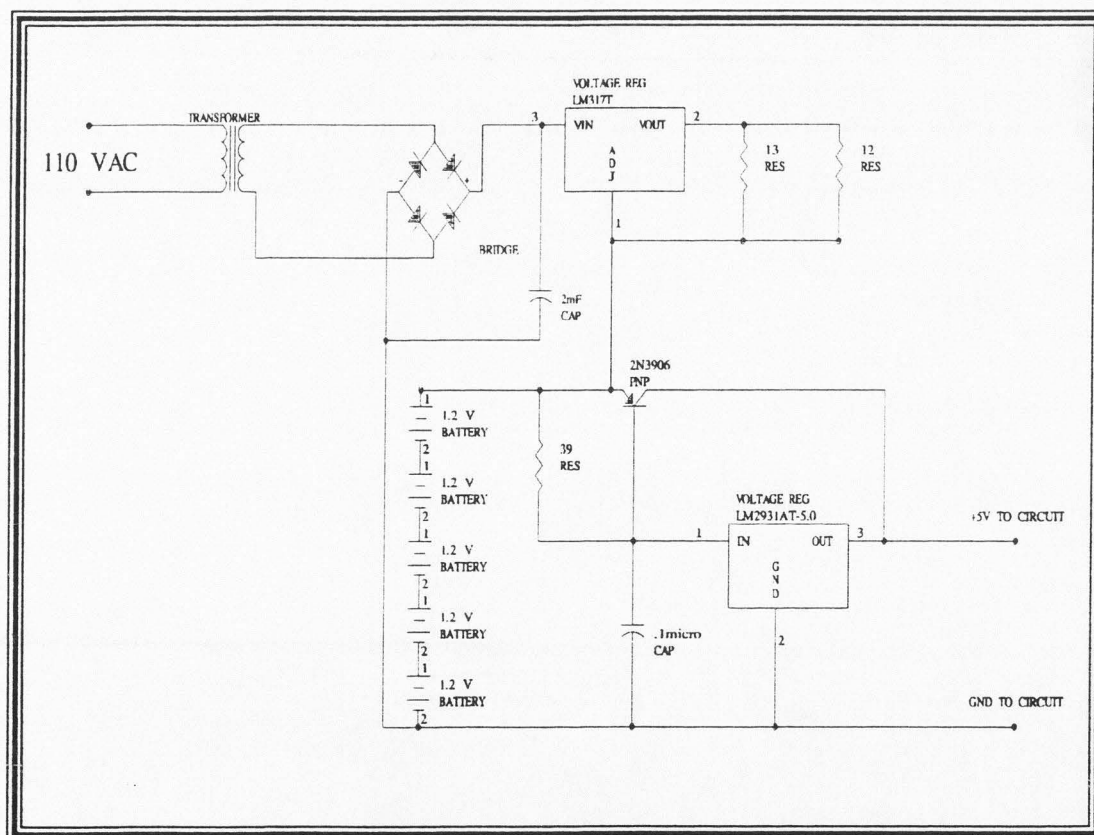


Figure 8. Rechargeable Power Supply

battery has a potential difference of 1.2 volts, making the five batteries in series total 6.0 volts. Charging circuitry is also included for recharging the batteries when they become discharged. The charging circuitry is also shown in Figure 8. The transformer shown plugs directly into any household 110 vac outlet. This rectifying

transformer converts 110 vac to 12 vdc. This is attached directly to a LM317T voltage regulator, which is being used as a constant current source that supplies 200 milliamps to charge the batteries. The batteries have a capacity of 2 amp hours, so the 200 milliamp current provides a standard charging rate of approximately one tenth the rated hour capacity. This means that 14-16 hours of charge time is required to fully charge the batteries.

Results and Recommendations

Talk4Me was evaluated by Beth Foley, Associate Professor of Communicative Disorders, who works constantly with individuals who have a wide range of speaking disabilities. Beth felt Talk4Me would be extremely useful in school settings with both young children who are developing language skills as well as those with developmental disabilities (e.g., mental retardation, Down syndrome). She also felt Talk4Me could prove very beneficial for "... adult users with short-term and/or relatively limited communication device needs (e.g., laryngectomees, stroke patients)." She continues, "Talk4Me fills this gap in the market, and should provide an effective, affordable communication system for a variety of individuals with severe speech impairments. An additional feature which makes Talk4Me an attractive new option is the scanning option, which enables even the most severely physically impaired user to access the system successfully via a single switch. . . . Talk4Me fills a long-standing need in the AAC (augmentative and alternative communication) system market for a versatile, high quality, reasonably priced AAC device." Beth also included recommendations to improve the Talk4Me communication aid. Chief of these recommendation was a sturdier, more streamlined case with a handle for ease of carrying. Refer to the appendix for Beth Foley's complete evaluation.

The initial prototype of Talk4Me was convenient to use and was able to record and play back messages as desired. Talk4Me also proved to be easy to fully reprogram and/or make minor changes in the programmed messages.

The success of Talk4Me has spawned a joint effort by the Center for Persons with Disabilities, Sunshine Terrace, and the Electrical Engineering department at USU to design and build a production quality Talk4Me. This next phase in the development of Talk4Me will include the following added features for improved performance:

- 1) more durable and reliable membrane switches
- 2) fully developed and tested rechargeable battery system
- 3) more durable and streamlined case
- 4) multiple level programing options
- 5) improved software control routines
- 6) printed circuit board
- 7) external switch hook up to control row column scanning
- 8) multiple memory configurations
- 9) improved message queuing system (switch identifiers)

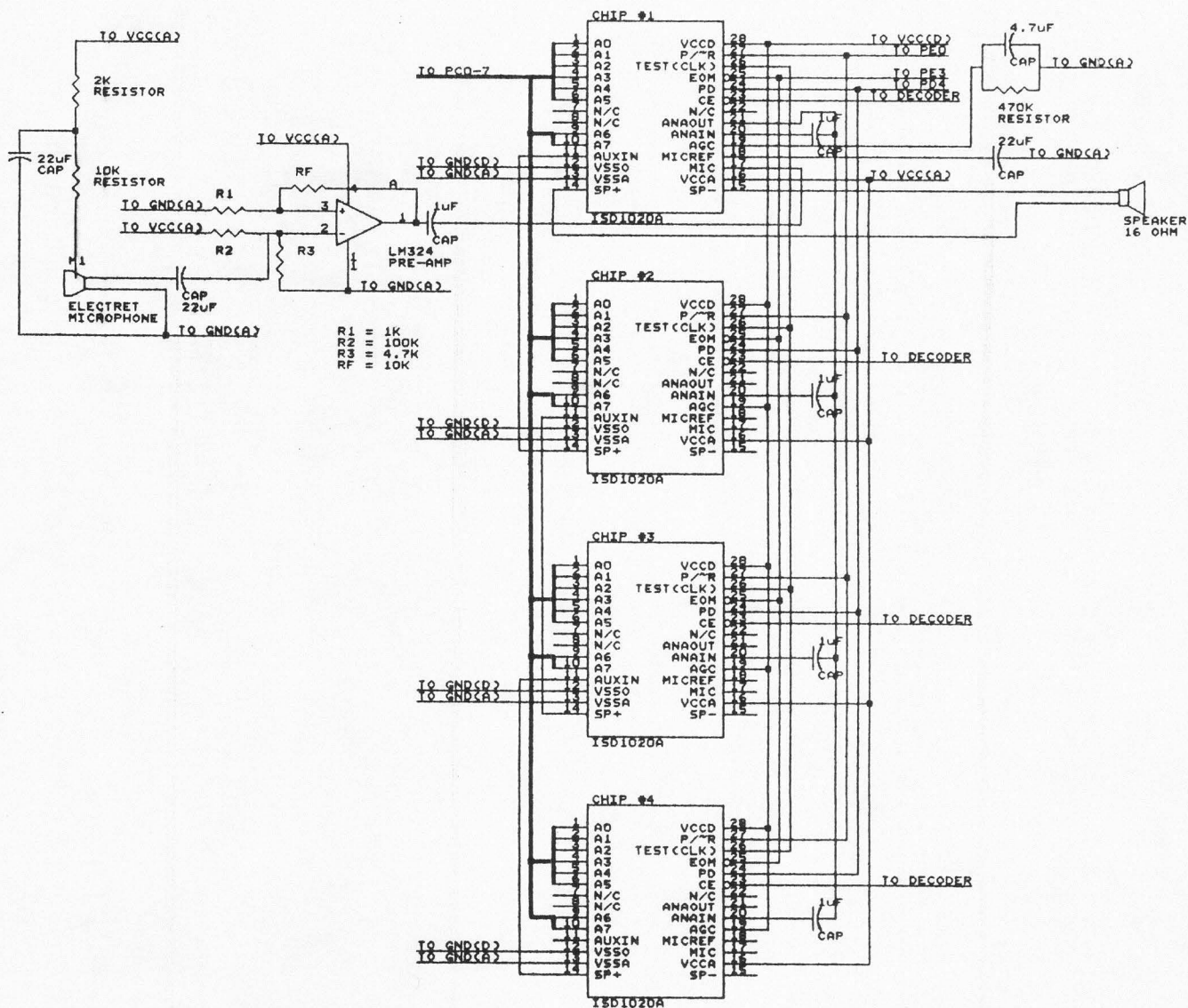
With these added features Talk4Me will be a significant addition to the devices which work to improve the quality of life for both individuals with severe speech impairments and all those who associate with them.

Appendix

Evaluation by Beth Foley, Associate Professor of Communicative Disorders.

The Talk4Me augmentative communication device represents a much needed low cost addition to current AAC systems market. There are no existing devices in the \$500 - 1000 price range that provide such high quality speech output and ease of use. The Talk4Me is appropriate for a variety of AAC users with a range of communication needs. For example, because it is so easy to program, Talk4Me would be extremely useful in school settings with young children who are developing language skills and require frequent changes and/or expansions in the vocabulary available on their communication devices. It would also be especially effective for students with developmental disabilities (e.g., mental retardation, Down Syndrome) who need highly intelligible speech output in order for them to understand it and/ or be understood by their peers. Adult users with short-term and/or relatively limited communication device needs (e.g., laryngectomees, stroke patients) would also benefit greatly from this technology. Currently, devices in the \$500-1000 range utilize low quality speech technology (e.g., the Echo speech synthesizer) which is robotic sounding and often unintelligible to young children and/or individuals with significant deficits. Talk4Me fills this gap in the market, and should provide an effective, affordable communication system for a variety of individuals with severe speech impairments. An additional feature which makes Talk4Me is the scanning option, which enables even the most severely physically impaired user to access the system successfully via a single switch. In addition, Talk4Me provides a significantly greater storage capacity than comparable communication devices using digitized speech technology. Talk4Me fills a long-lasting need in the AAC (Augmentative and Alternative Communication) system market for a versatile, high quality, reasonably priced AAC device.

The Talk4Me prototype could be improved in a number of ways. First, the current case used to house the device is bulky and larger than it needs to be. A sturdier, more streamlined case with a handle for ease of carrying would make the device more attractive, more durable, and more portable. The overlay which will be used to display communication symbols needs to be more durable and resistant to spills, drooling, etc. The surface of the display must be such that the communication symbols can be added and removed with relative ease.



Darren Blaser and Scott Sorenson	
Title	Audio Subsystem
Size	Document Number
B	1
Date:	July 22, 1993 Sheet 1 of 1

```

*****
*****      AUGMENTATIVE COMMUNICATIONS DEVICE      *****
*****      DARREN BLASER AND SCOTT SORENSON        *****
*****      PROGRAM FOR THE MOTOROLA                *****
*****      M68HC811E2 MICROCONTROLLER             *****
*****

```

```

*****
*****      INITIALIZATION BLOCK                    *****
*****

```

```

porta    equ    $00
portb    equ    $04
portc    equ    $03
portd    equ    $08
porte    equ    $0A

```

```

*****
*****      initialize variables                    *****
*****
      org    $F800      start variable table at beginning
*              of EEPROM

```

```

keytab    fcb    $00,$28,$50,$78,$00,$28,$50,$78
*          rows 0 and 1 config#1
      fcb    $00,$28,$50,$78,$00,$28,$50,$78
*          rows 2 and 3
      fcb    $00,$09,$12,$1B,$24,$31,$3E,$4B,$58,$68
      fcb    $78,$88,$98,$B2,$C8,$E2
      fcb    $00,$10,$20,$30,$40,$50,$60,$70,$80,$A0
      fcb    $C0,$E0,$ff,$ff,$ff,$ff
      fcb    $00,$09,$12,$1B,$24,$2D,$36,$3F,$48,$55
      fcb    $62,$6F,$7C,$89,$A4,$C0

```

```

chptab    fcb    $00,$00,$00,$00,$01,$01,$01,$01
*          rows 0 and 1 config#1
      fcb    $02,$02,$02,$02,$03,$03,$03,$03
*          rows 2 and 3

```

```

lights    fcb    %11100001,%11010001,%10110001,%01110001
*          row 0 lights
      fcb    %11100010,%11010010,%10110010,%01110010
*          row 1 lights
      fcb    %11100100,%11010100,%10110100,%01110100
*          row 2 lights
      fcb    %11101000,%11011000,%10111000,%01111000
*          row 3 lights

```

```

row#1     fcb    %00000001
row#2     fcb    %00000010
row#3     fcb    %00000100
row#4     fcb    %00001000

```



```

times      org    $f8a0
           fcb    $40,$40,$40,$ff,$21,$21,$21,$ff  rows 0 and 1
           fcb    $16,$16,$16,$ff,$10,$10,$10,$ff  rows 2 and 3
           org    251          put variables at top of ram
count      rmb    1
key        rmb    1          variables for scan routine
col        rmb    1
config     rmb    1          config stores value for
                             configuration table
adr        rmb    1

```

```

*****
*****          start program !!          *****
*****

```

```

*          org    $F900          start code 100 bytes from start of
start      EEPROM

```

```

*****          INITIALIZE PORTS FOR INPUT/OUTPUT          *****
*****          port a (located at $1000)          *****

```

```

*   pa0 keyboard matrix read line #1
*   pa1 keyboard matrix read line #2
*   pa2 keyboard matrix read line #3
*   pa3 keyboard matrix read line #4
*   pa4 keyboard matrix write line #1
*   pa5 keyboard matrix write line #2
*   pa6 keyboard matrix write line #3
*   pa7 memory configuration (low bit)

```

```

*****          port b (located at $1004)          *****
*   pb0-7 control for keyboard LED matrix (all output)

```

```

*****          port c (located at $1003)          *****
*   pc0-7 address bus to all ASA chips (all output)
*          program port c as output port
*          0=input 1=output

```

```

ldx    #$1000
ldaa   #$FF          load 11111111 into accumulator a
staa   $7,x          program port c

```

```

*****          port d (located at $1008)          *****

```

```

*   pd0 |
*   pd1 | >to 138 3 to 8 multiplexer for /ce's
*   pd2 | (ASA chip 1-4 == bit address 0-3)
*          *note address 5 disenables all ASA chips
*   pd3 keyboard matrix write line #4
*   pd4 power down for all ASA chips
*   pd5 memory configuration (high bit)
*   pd6 ?
*   pd7 ?

```

```

*   program port d as output for pins pd0-4

```

```

        ldaa #$1F          load 00011111 into accumulator a
        staa $9,x          program port d
        ldaa #%00010111
        staa $8,x          initialize port d

*****      port e (located at $100A)      *****
*   pe0      play /record          * note all inputs
*   pe1      memory overrun
*   pe2      direct select /scan
*   pe3      /EOM signal from all chips
*   pe4-7    does not exist on 48 pin dip package

*****      INITIALIZE STACK      *****

        LDS      #250          points almost to top of ram

*****
*****      Configuration Initialization      *****
*****      Read configuration from input dip switch *****
*****      and store configuration in variable config *****
*****

        ldaa #%00100000      mask of bit on port d to test
        anda portd,x
        bne high            brach if high bit set

        ldaa #%10000000      mask of bit on port d
        anda porta,x
        bne one             brach if low bit set

        ldaa #$00            in config %00 (zero)
        bra mread

one   ldaa #$01            in config %01 (one)
        bra mread

high  ldaa #%10000000      check low bit for value
        anda porta,x
        bne three          in config %11 (three)

        ldaa #$02            in config %10 (two)
        bra mread
three ldaa #$03

mread      staa config      store configuration in variable
*          (config)

*****
*****
*****      Begin main modules of program      *****
*****
*****      Read mode control and jump to appropriate address *****
*****

```

```

* Read mode control data from port e(0-4)
* 0 - play /record
* 1 - memory overrun
* 2 - Direct Select /Scan
* 3 - Memory Config (low bit)

```

```

mode ldaa #$01          test input e0 for /record
      anda porte,x
      beq  record       branch to record mode

```

```

* brclr #$100A #$04 scan BRANCH TO SCAN MODE
* note scan mode not enabled yet default to dselect mode
* brset #$100A #$04 dselect BRANCH TO D.S. MODE

```

```

*****
*****          Direct Select Module          *****
*****

```

```

      bsr  scanh          branch to scan subroutine
      ldaa #$ff           check for no key pressed
      cmpa key            $ff value for no key press
      beq  mode           no key pressed -- check mode
*                               again
      bclr portd,x 0x00010000 power up all ASA chips
      bsr  address       put appropriate address on bus
      bsr  waitee        wait for eeprom to write
      bsr  chipenable    enable first chip
      bsr  waitee
      bset portd,x 0x00000111 chip disable
      ldaa 0x00001000      read /EOM signal
eom  anda porte,x
      tsta                test for end of message
      bne  eom
      bset portd,x 0x00010000 power down all ASA chips
      bra  mode           end of dselect module

```

```

*****
*****          Recording Module          *****
*****

```

```

record  bsr  scanh          branch to scan subroutine
        ldaa #$ff           check for no key pressed
        cmpa key            $ff value for no key press
        beq  mode           no key pressed -- check mode
*                               again
        bclr portd,x 0x00010000 power up all ASA chips
        bsr  address       put appropriate address
on bus
        bsr  waitee        wait for eeprom to write
*                               turn on recording light
        ldy  $f850         load light offset into y
reg
        ldab key           load acc b key offset
        aby                add offset (acc b) with index reg y

```



```

*                                and place result in index reg y
                                load light output into acc a
                                turn on light
                                enable proper chip (start
*                                recording)
                                wait until key is
*                                released
                                chip disable (stop
*                                recording)
                                wait ???
                                power down all ASA chips
samhold ldaa #00001000 read /EOM signal (EOM)
                                anda porte,x
                                tsta test for end of chip
*                                message

```

```

                                beq out
*                                (check for same key)
                                ldaa key load key pressed into acc a
                                psha save a
                                bsr scan check for same key press
                                ldab key
                                pula
                                cba compare previous with present
                                bne out
                                ldaa $#ff
                                cmpa key
                                beq out
                                bra samhold
                                out bra mode end of record module

```

```

*****
scan1 bra scan help for scan subroutine to scan
keyboard
*****

```

```

*****
***** Subroutines *****
*****
*****

```

```

*****
***** wait for eeprom to write *****
*****

```

```

waitee ldy #10000
del dey
bne del
rts

```

```

*****
***** wait ca 10 msec *****
*****

```



```

* wait10 ldy #300000
* address help
address      jmp  addr8
chipenable   jmp  enable
scanh        jmp  scan

```

```

*****
*****                      same key                      *****
*****                      (stop recording)                *****
*****                      insert possible delay ??        *****
*****

```

```

samekey      ldaa #$00          start counter at zero (timer)
              staa count

```

```

same         ldaa #%00001000    read /EOM signal (EOM)
              anda porte,x
              tsta              test for end of chip message
              beq  timeout

```

```

*            (check for same key)
              ldaa key          load key pressed into acc a
              psha             save a
              bsr  scan         check for same key press
              ldab key
              pula
              cba              compare previous with present
              bne  timeout

```

```

              inc  count        (test for time expiration)
*            ldy  #$ff          delay for smoother time counting
*delay      dey
*            cpy  #$00
*            bne  delay

```

```

              ldy  #$f8a0       offset for time table
              ldab key          load acc b with complete offset
              aby              add offset (acc b) with index reg y
*            and placc result in index reg y
              ldaa $00,y        load timeout # into acc b
              cmpa count
              bhs  same         ie if accb=> count branch

```

```

timeout      ldaa $f0          turn off recording light
              staa portb,x
              rts
*            waitee help
waite        jmp  waitee

```

```

*****
*****                      scan module!!                  *****
*****

```

```

* subroutine that scans keyboard matrix
* input expected:  register x = $1000
* output returned: # of button depressed (0-15) = memory

```

```

*               location <key>
* if no button is depressed $FF is returned in key
*****
*   note:  rows    = outputs (four most significant bits of
*               port a)
*   columns = inputs (four least significant bits of
*               port a)
*****

*   initialize port a

scan ldaa #$00           initialize all bits on porta to 0
    staa porta,x
    bclr portd,x    #%00001000    clr write line #4
    bsr  waitee     wait for eeprom to write

row0    bset      porta,x    #%00010000    set row0 bit
    bsr  waitee     wait for eeprom to write
    ldab  #$00      load row #(00) into accb
    ldaa  #$0f      read all four inputs
    anda  porta,x
    tsta
    bne  col0       (possibly removable instruction)
                    row #0 active find column

row1 bclr porta,x    #%00010000    clear row 0 bit
    bsr  waitee
    bset porta,x    #%00100000    set row1 bit
    bsr  waitee
    ldab  #$01      load row #(01) into accb
    ldaa  #$0f
    anda  porta,x
    tsta
    bne  col0       row #1 active find column

row2 bclr porta,x    #%00100000    clear row 1 bit
    bsr  waitee
    bset porta,x    #%01000000    set row 2 bit
    bsr  waitee
    ldab  #$02      load row #(02) into accb
    ldaa  #$0f
    anda  porta,x
    tsta
    bne  col0       row #2 active find column

row3 bclr porta,x    #%01000000    clear row 2 bit
    bsr  waitee
    bset portd,x    #%00001000    set row 3 bit
    bsr  waitee
    ldab  #$03      load row #(03) into accb
    ldaa  #$0f
    anda  porta,x
    tsta
    bne  col0       row #3 active find column

```

bra notouch	all rows inactive
* address help	
addrs bra addrs1	
enable bra cenable	
col0 ldaa #\$00	record the col number
staa col	in variable col
ldaa #%00000001	check to see if col 0 is active
anda porta,x	
tsta	
bne calc	col 0 active branch to key
calculations	
col1 ldaa #\$01	record the col number
staa col	in variable col
ldaa #%00000010	check to see if col 1 is active
anda porta,x	
tsta	
bne calc	col 1 active branch to key
calculations	
col2 ldaa #\$02	record the col number
staa col	in variable col
ldaa #%00000100	check to see if col 2 is active
anda porta,x	
tsta	
bne calc	col 2 active branch to key
calculations	
col3 ldaa #\$03	record the col number
staa col	in variable col
ldaa #%00001000	check to see if col 3 is active
anda porta,x	
tsta	
bne calc	col 3 active branch to key
calculations	
bra notouch	no columns are active
calc ldaa #\$04	multiplying the row (in acum b)
mul	with 4 and the result in
acc D	
ldaa col	loading col in acc. b
aba	adding the accs a and b
staa key	storing the result in key
bra return	
notouch ldaa \$FF	load \$ff into key
staa key	
return rts	return from scan subroutine

*****	write address subroutine *****


```

*****
*           Getting the appropriate address from the
*           configuration (key) table.
*
* expected inputs:  config contains configuration (0-3)
*                   key contains key # pressed (0-15)
*
* outputs:         correct address for depressed key is
*                   written to port c (address bus)
*****

```

```

addrsl      ldaa #$00 (delete when config enabled)
            staa config (delete when config enabled)
            ldaa config loading configuration into acc a
            ldab #$10 loading 16 into acc b
            mul multiply acca and accb and store the
*            result in a
*            the value in acca now contains the
*            sectional offset for the address

            ldaa key load key # into acc b
            aba adding key offset and sectional offset
            staa adr and store the result in adr

            ldy #$f800 load keytab offset into y reg
            ldab adr load acc b with complete offset
            aby add offset (acc b) with index reg y
*            and place result in index reg y

            ldaa $00,y load address into acc a
            staa portc,x write address to portc
            rts

```

```

*****
***** chipenable routine *****
*****

```

```

cenable     ldaa #$00
            staa config
            ldaa config loading configuration into acc a
            ldab #$10 loading 16 into acc b
            mul multiply acca and accb and store the
*            result in a
*            the value in acca now contains the
*            sectional offset for the address

            ldaa key load key # into acc b
            aba adding key offset and sectional offset
            staa adr and store the result in adr
            ldy #$f840 load chiptab offset into y reg
            ldab adr load acc b with complete offset
            aby add offset (acc b) with index reg y
*            and place result in index reg y

            ldab $00,y load chip # into acc a

```



```
ldaa #%11111000  save current configuration on port d
anda portd,x      except for chip enable info
aba
staa portd,x      enable chip
rts
```

```
*****
*****                               End of Progam                               *****
*****
```

```
org  $FFFE
fdb  start
end
```

```
*****          note:  nice delay routine
* ldaa          #10
* tipsy         ldy  #$ffff
* wait2         dey
* cpy           #$00
* bne           wait2
* deca
* tsta
* bne           tipsy
```