PROGRAM/HARDWARE DESIGN

Part 1: Program Components

A. HARDWARE:

48 key keyboard built from 4−key components, the logic is described below:

We periodically turn on the power line for each row, and check which of the 12 columns are "on", say we detect a signal 0b1000 0000 0000 for the column when we have just powered up the first row, then we are sure that the 1st row, 1st column key has been pressed.

To obtain a representive ASCII character from a keypress:

1. We transform the row number and column number into a 8 bit keycode, upper 4 bits holds the row number and lower 4 bits holds the column number
2. From this keycode, we obtain its integer key number (0−48) from a lookup similar to that used by the lab keyboard
3. We stored in memory the ASCII character in order with the keynumber, so to find say ASCII represented by keynumber 5, we go to the memory location start + offset 5. This is more efficient linear time lookup.

We modulated all keyboard functions to 3 subroutines: GetKey, BtnNum and Num2Char. We modulated LCD functions to other helpful subroutines such as printChar, printFlash, printRam.

B. SOFTWARE:

1. Storing English−Malay pairs in Flash ˇ We parsed a text file containing the dictionary into a format that the assembly file can read from using Java
2. Recording user entries ˇ Integrated with our keyboard code, all keyboard entires are recorded and parsed.
3. String processing ˇ We wrote useful subroutines such as string compare, string copy, etc.
4. Display ˇ We manipulated memory addresses and keyboard polling to achieve the "Scroll forward", "Scroll backward" fucntionality
5. Implementing the Hangman game logic ˇ The hangman games require us to know what the user just guessed and what he has guessed before. We implemented this using a "mask". The "mask" has the same amount of space as the "target" word, except that it is initialized to be '_' 's and each time user has a search hit, the corresponding characters will replace the placeholders. So say the original word is "cat", if user guessed "c" the first time, and "t" the second time, mask will become "c_ _" and then "c _ t". "Mask" is traversed the same time the "Target" is traversed. We then display "Mask" to the LCD screen. This is the most memory and computation efficient algorithm we have devised.
6. Randomizing ˇ We randomly pick words from our database. To get a random number

we leave TIMER1 ticking and we used its high and low bytes for the address offsets.

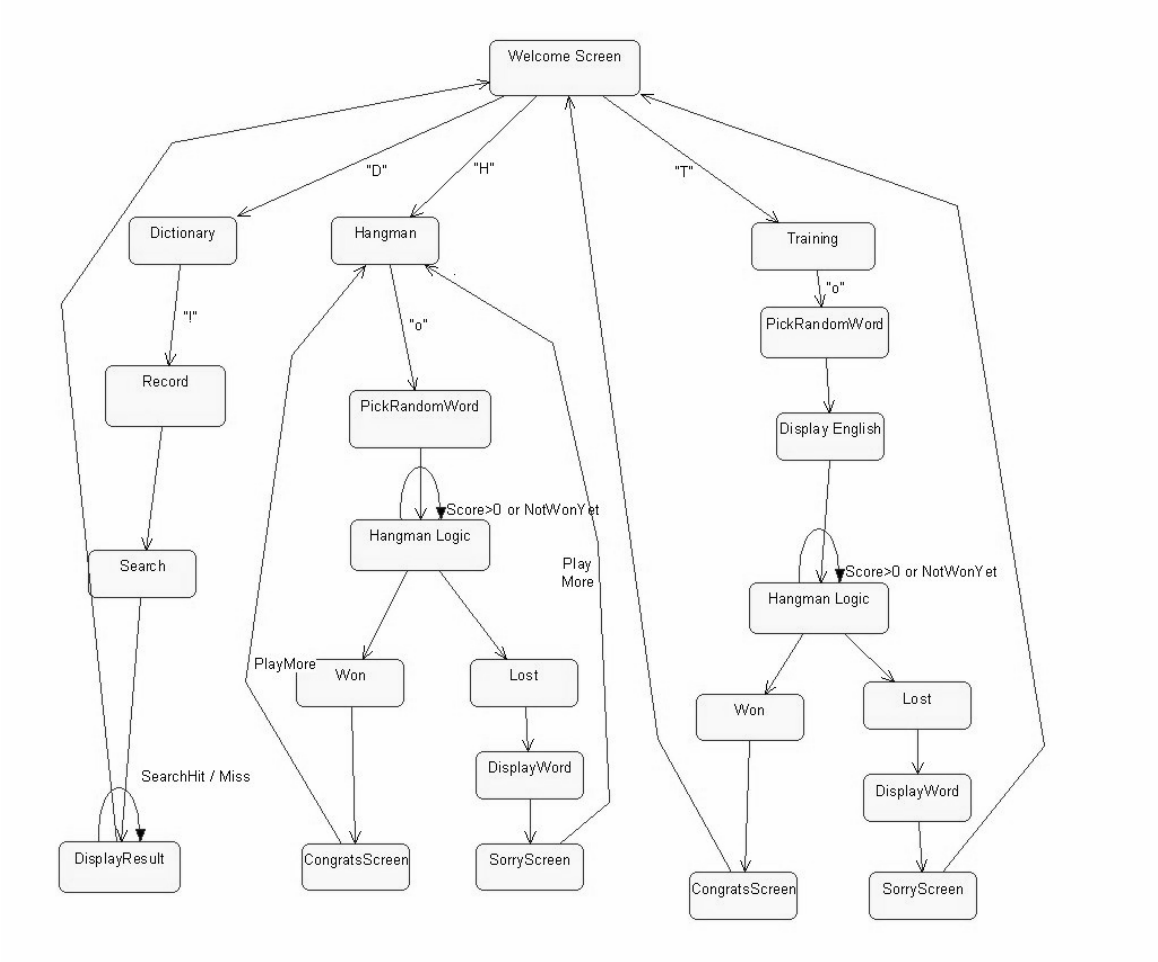
Part 2: Results of the design −− speeds, accuracy, usability, etc.

The Dictionary performance is very reasonable. No humans can detect the latency between pressing "enter" and the appearance of English words on LCDs. We have made an effort to make the UI intuitive and easy to use with limited resources (16 char LCD), and believe that we have obtained the best that we can. The Dictionary, hangman have almost perfect accuracy, except that at rare occasions there can be parsing errors, or memory going out of bound.

Part 3: In the future...

1. We tried using EEPROM instead of Flash to store the Malay−English pairs but we obtained a Serial EEPROM instead of a parallel one, which is technically more difficult to program. A technically more feasible option is a EEPROM programmer, a parallel EEPROM or external SRAM.
2. A more advanced search algorithm, such as a hashtable
3. A more comfortable keyboard

Part 4: Schematic Flowchart



Flowchart of the overall logic

Part 5: Code

COMMONLY USED MACROS FOR AT90S4414/8515

```
.include "8515def.inc"
.device at90s8515

;BEGIN BRANCH MACROS-------------------------
;skips next @0 instructions if equal (=ifne)
.macro      skipe
.set        _skipe = PC + 1 + @0
            breq        _skipe
.endmacro

;performs next @0 instructions if not equal (=skipe)
.macro      ifne
.set        _ifne = PC + 1 + @0
            breq        _ifne
.endmacro

;skips next @0 instructions if not equal (=ifeq)
```

```
.macro      skipne
.set        _skipne = PC + 1 + @0
            brne        _skipne
.endmacro

;performs next @0 instructions if equal (=skipne)
.macro      ifeq
.set        _ifeq = PC + 1 + @0
            brne        _ifeq
.endmacro

;skips next @0 instructions
.macro      else
.set        _else = PC + 1 + @0
            rjmp        _else
.endmacro

;performs next @0 instructions if greater or equal (signed)
.macro      ifge
.set        _ifge = PC + 1 + @0
            brlt        _ifge
.endmacro

;performs next @0 instructions if same or higher (unsigned)
.macro      ifsh
.set        _ifsh = PC + 1 + @0
            brlo        _ifsh
.endmacro

;performs next @0 instructions if less than (signed)
.macro      iflt
.set        _iflt = PC + 1 + @0
            brge        _iflt
.endmacro

;performs next @0 instructions if lower than (unsigned)
.macro      iflo
.set        _iflt = PC + 1 + @0
            brsh        _iflt
.endmacro

;do loops
;used with whilenz
;@0 = loop number (for nested loops)
.macro do
.set        _do@0 = PC
.endmacro

;executes from the @1'th do if @0 is not zero
.macro whilenz
            tst         @0
            brne        _do@1
.endmacro

;executes from the @0'th do if zero flag is set
.macro      whilene
            brne        _do@0
.endmacro

;executes from the @0'th do if zero flag is not set
.macro      whileeq
            breq        _do@0
.endmacro

;executes from the @0'th do if greater or equal
.macro      whilege
            brge        _do@0
```

```
.endmacro

;executes from the @0'th do if less than
.macro      whilelt
            brlt        _do@0
.endmacro
;END BRANCH MACROS--------------------------

;TIMER MACROS-------------------------------
;Prescaler calculations:
;  1x up to 6.4E-5 (or 0.016384 w/Count)
;  8x up to 5.12E-4 (or 0.131072 w/Count)
;  64x up to 4.096E-3 (or 1.048576 w/Count)
;  256x up to 0.016384 (or 4.194304 w/Count)
;  1024x up to 0.065536 (or 16.777216 w/Count)
;
;  Formula: 0.25us x Prescaler x TCNT0_reset x Count = time
;    e.g.: 0.25us x 256 x 250 x 125 = 2 seconds
;  Pre-scalers:
;    cTimeX0 -- stop timer
;    cTimeX1 -- no pre-scale
;    cTimeX8 -- pre-scale by 8
;    cTimeX64 -- pre-scale by 64
;    cTimeX256 -- pre-scale by 256
;    cTimeX1024 -- pre-scale by 1024

.equ        cTimeX0 = 0
.equ        cTimeX1 = 1
.equ        cTimeX8 = 2
.equ        cTimeX64 = 3
.equ        cTimeX256 = 4
.equ        cTimeX1024 = 5

;setTim1 -- enables timer overflow interrupt 1 and
;       sets pre-scaler
;  @0 = Pre-scaler
;Destroys: r16
.macro      setTim1
            ldi         r16, exp2(TOIE1)
            out         TIMSK, r16
            ldi         r16, @0
            out         TCCR1B, r16
.endmacro

;setTim0 -- enables timer overflow interrupt 0 and
;           sets pre-scaler
;  @0 = Pre-scaler
;Destroys: r16
.macro      setTim0
            ldi         r16, exp2(TOIE0)
            out         TIMSK, r16
            ldi         r16, @0
            out         TCCR0, r16
.endmacro

;setTime -- sets the Timer Interrupt Mask Register (TIMSK)
;           (enables/disables timer interrupts)
;  Note: 0 to disable, 1 to enable
;  @0 - TOIE1 (Overflow Interrupt for Timer 1)
;  @1 - OCIE1A (Output CompareA Match Interrupt)
;  @2 - OCIE1B (Output CompareB Match Interrupt)
;  @3 - TICIE1 (Input Capture Interrupt)
;  @4 - TOIE0 (Overflow Interrupt for Timer 0)
;  @5 - Prescaler 0
;  @6 - Prescaler 1
;Destroys: r16
.macro      setTime
```

```
            ldi         r16, TOIE1*@0 + OCIE1A*@1 + OCIE1B*@2 +
TICIE1*@3 + TOIE0*@4
            out         TIMSK, r16
            ldi         r16, @5
            out         TCCR0, r16
            ldi         r16, @6
            out         TCCR1B, r16
.endmacro

;clrTime -- disables all timers
;Destroys: r16
.macro      clrTime
            clr         r16
            out         TIMSK, r16
.endmacro

;clrTim1 -- disables timer overflow interrupt 1
;Destroys: r16
.macro      clrTim1
            cbi         TIMSK, TOIE1
.endmacro

;clrTim0 -- disables timer overflow interrupt 0
;Destroys: r16
.macro      clrTim0
            cbi         TIMSK, TOIE0
.endmacro

;END TIMER MACROS----------------------------

;ANALOG COMPARATOR MACROS--------------------

;setAC -- set-up analog comparator
; @0 - Input Capture enable? (0/1)
; @1 - ACToggle, ACFall, ACToggle
;   ACToggle - Comparator Interrupt on Output Toggle
;   ACFall - Interrupt on Falling Output Edge
;   ACRise - Interrupt on Rising Output Edge
.macro      setAC
            ldi         r16, ACIE + ACIC*@0 + @1
            out         ACSR, r16
.endmacro

.equ        ACToggle = 0
.equ        ACFall = 2
.equ        ACRise = 3

;clrAC -- disables Analog Comparator, and reset
;  all parameters in ACSR to zero (default)
.macro      clrAC
            clr         r16
            out         ACSR, r16
.endmacro

;END ANALOG COMPARATOR MACROS----------------

;EXTERNAL INT MACROS-------------------------
;setExt -- enables external interrupt
;  @0 = Sense Control
;  Sense Controls:
;    cExtLevel -- Level-sensitive
;    cExtFall -- Negative-edge triggered
;    cExtRise -- Positive-edge triggered
;Destroys: r16
.macro      setExt
            ldi         r16, exp2(INT0)
            out         GIMSK, r16
```

```
            ldi         r16, @0
            out         MCUCR, r16
.endmacro
.equ        cExtLevel = 0
.equ        cExtFall = 2
.equ        cExtRise = 3

;clrExt -- clears external interrupt
;Destroys: r16
.macro      clrExt
            clr         r16
            out         GIMSK, r16
.endmacro
;END EXTERNAL INT MACROS---------------------

;PORT MACROS---------------------------------
;performs next @2 instructions if bit clear (button pressed)
;@0 = PINA OR PINB OR PINC OR PIND
;@1 = Pin #
;@2 = # of instructions to perform
.macro      ifbc
.set        _ifbc = PC + 2 + @2
            sbic        @0, @1
            rjmp        _ifbc
.endmacro

;performs next @2 instructions if bit set (button released)
;@0 = PORTA or PORTB or PORTC or PORTD
;@1 = Pin #
;@2 = # of instructions to perform
.macro      ifbs
.set        _ifbs = PC + 2 + @2
            sbis        @0, @1
            rjmp        _ifbs
.endmacro
;END PORT MACROS-----------------------------

;UART MACROS---------------------------------
;setUART -- setup UART
; TXempty, RXdone, RX and TX
;  @0 - baud rate setting for UBRR (page 40)
;Destroys: r16
.macro      setUART
            ldi         r16, 0b10011000
            out         UCR, r16
            ldi         r16, @0
            out         UBRR, r16
.endmacro

;ramUART -- load data at RAM address to UDR/UART
;  @0 - RAM address
;Destroys: r16
.macro      ramUART
            ldi         ZL, LOW(@0)
            ldi         ZH, HIGH(@0)
            ld          r16, Z
            out         UDR, r16
.endmacro

;fshUART -- load data at flash address to UDR/UART
;  @0 - flash address
;Destroys: r16
.macro      fshUART
            ldi         ZL, LOW(@0 *2)
            ldi         ZH, HIGH(@0 *2)
            ld          r16, Z
            out         UDR, r16
```

```
.endmacro
;END UART MACROS-----------------------------

;EEPROM MACROS-------------------------------
;reads ROM indirect
;  @0 - register with ROM byte
;  @1 - label/address
;Destroys: r16
.macro      rdROMi
            clr         r16
            out         EEARH, r16
            ldi         r16, @1
            out         EEARL, r16
            sbi         EECR, EERE
            in          @0, EEDR
.endmacro

;reads ROM
;  @0 - register with ROM byte
;  @1 - register with label/address
;Destroys: r16
.macro      rdROM
            clr         r16
            out         EEARH, r16
            out         EEARL, @1
            sbi         EECR, EERE
            in          @0, EEDR
.endmacro
;END EEPROM MACROS---------------------------

;MISC MACROS---------------------------------
.macro      halt
_halt:      rjmp        _halt
.endmacro

;setup1 RESET, EXT_INT1, EXT_INT2, T1_CAPTURE, T2_CAPTURE,
;setup2     T1_COMPARE, T2_COMPARE, T1_OVERFLOW, T0_OVERFLOW, SPI_STC,
;setup3 Rx_COMPLETE, Tx_COMPLETE, ANA_COMP
.macro      setup1
            rjmp        @0
            rjmp        @1
            rjmp        @2
            rjmp        @3
            rjmp        @4
            rjmp        @5
.endmacro

.macro      setup2
            rjmp        @0
            rjmp        @1
            rjmp        @2
            rjmp        @3
            rjmp        @4
            rjmp        @5
.endmacro

.macro      setup3
            rjmp        @0
            rjmp        @1
            rjmp        @2
.endmacro

;Sets up stack pointer
;Destroys: r16
.macro      setStack
            ldi         r16, LOW(RAMEND)
            out         SPL, r16
```

```
            ldi         r16, HIGH(RAMEND)
            out         SPH, r16
.endmacro

;setDD -- set data direction
;  @0 = DDRB ('1'=output pin, '0'=input)
;  @1 = DDRD ('1'=output pin, '0'=input)
;  @2 = DDRA ('1'=output pin, '0'=input)
;  @3 = DDRC ('1'=output pin, '0'=input)
;Destroys: r16
.macro      setDD
            ldi         r16, @0
            out         DDRB, r16
            ldi         r16, @1
            out         DDRD, r16
            ldi         r16, @2
            out         DDRA, r16
            ldi         r16, @3
            out         DDRC, r16
.endmacro

;flashZ     -- load flash address to Z
;  @0 - flash address
;Output: ZL gets LOW(@0 * 2)
;        ZH gets HIGH(@0 * 2)
.macro      flashZ
            ldi         ZL, LOW(@0 * 2)
            ldi         ZH, HIGH(@0 * 2)
.endmacro

;ramZ -- load RAM address to Z
;  @0 - RAM address
;Output: ZL gets LOW(@0)
;        ZH gets HIGH(@0)
.macro      ramZ
            ldi         ZL, LOW(@0)
            ldi         ZH, HIGH(@0)
.endmacro

;END MISC MACROS-----------------------------
```

KEYBOARD INTERFACE CODE

The following is the code we used to interface with the keyboard we built.

```
;-------------------------------------
;KB48MAC.ASM -- 12x4 Keyboard
;
;All labels and variables starts with prefix:
;               k48_
;
;INPUT:
;  constants to be defined:
;               k48_PORT1; the port used for controlling the keyboard (.e.g .equ k48_PORT1=PORTA)
;               k48_PORT2; the port used for controlling the keyboard (.e.g .equ k48_PORT2=PORTD)
;               k48_PIN1 ; the pin used for controlling the keyboard (e.g. .equ k16_PIN1=PINA)
;               k48_PIN2 ; the pin used for controlling the keyboard (e.g. .equ k16_PIN2=PIND)
;               k48_DD1  ; for setting the Data-Direction (e.g. .equ k48_DD1=DDRA)
;               k48_DD2  ; for setting the Data-Direction (e.g. .equ k48_DD2=DDRD)
;  registers we use:
;               rLine
;  internal function:
;               checkLine


;Quick Reference
;---------------
;1. k48_kbhit ; checks if a key is pressed
;               INPUT: none
;               OUTPUT: r16 gets 0xFF if no key pressed or keycode if a key is pressed
;2. k48_getKey; waits until a key is pressed
;               INPUT: none
;               OUTPUT: r16 gets keycode
;3. k48_btnNum; translate keycode to button number
;               INPUT: r16 = keycode
```

```
;                OUTPUT: r16 = button number
;4. k48_Num2Char: translate keycode to ASCII character
;                INPUT: r16 = keynumber
;                OUTPUT: r16 = corresponding ASCII character




;---------------BEGIN k48_Num2Str-----------------------------------------------
;k48_Num2Char; translate keycode to button number
;INPUT: r16 = button number
;OUTPUT: r16 = ASCII character
;                +-------------------------+
;                | 0 1 2 3 4 5 6 7 8 9 o ? |
;       | Q W E R T Y U I O P X ! |
;       | A S D F G H J K L X X * |
;       | Z X C V B N M X X X X # |
;       +-------------------------+
k48_Num2Char:
                push            ZL              ;save registers before destroying...
                push            ZH
                push            r17
                clr             r17
                ;search the table for a match to the raw key code
                ;and exit with a button number
                flashZ          k48_keytASCII   ;table pointer in FLASH
                add             ZL, r16
                adc             ZH, r17

k48_Num2StrGet:
                lpm                             ; get the table entry

                mov             r16,r0
                pop             r17
                pop             ZH
                pop             ZL
                ret


;ASCII representations
k48_keytASCII:   .db '?', 'X', '9', '8', '7', '6', '5', '4', '3', '2', '1', '0'
                                .db '!', 'O', 'P', 'O', 'I', 'U', 'Y', 'T' ,'R', 'E', 'W', 'Q'
                                .db '*', 'X', 'X', 'L', 'K', 'J', 'H', 'G', 'F', 'D', 'S', 'A'
                                .db '#', 'X', 'X', 'X', ' ', 'M', 'N', 'B', 'V', 'C', 'X', 'Z'
-------------------------------------------------------------------------------


;---------------BEGIN k16_btnNum------------------------------------------------
;k48_btnNum; translate keycode to button number
;INPUT: r16 = keycode
;OUTPUT: r16 = button number
;       +-------------------------+
;       | 0 1 2 3 4 5 6 7 8 9 X X |
;       | Q W E R T Y U I O P X X |
;       | A S D F G H J K L X X X |
;       | Z X C V B N M X X X X X |
;       +-------------------------+
k48_btnNum:
                push            ZL              ;save registers before destroying...
                push            ZH
                push            r17
                ;search the table for a match to the raw key code
                ;and exit with a button number
                ldi             ZL, low(k48_keytbl*2)           ;table pointer in FLASH
                ldi             ZH, high(k48_keytbl*2)
                clr             r17

k48_bn1:
                lpm                             ; get the table entry
                cp              r16, r0         ; match?
                ifne            10
                                inc             r17
                                cpi             r17, 48         ;if not match still then illegal
                                ifeq            5               ;flag illegal by setting t-bit
                                                pop             r17
                                                pop             ZH
                                                pop             ZL
                                                set
                                                ret
                                adiw            ZL, 1
                                rjmp            k48_bn1
                mov             r16, r17
                clt
                pop             r17
                pop             ZH
                pop             ZL
                ret


;keyboard scancode
k48_keytbl:     .db 0x11,0x12,0x13,0x14,0x15,0x16,0x17,0x18,0x19,0x1A,0x1B,0x1C
                .db 0x21,0x22,0x23,0x24,0x25,0x26,0x27,0x28,0x29,0x2A,0x2B,0x2C
                .db 0x31,0x32,0x33,0x34,0x35,0x36,0x37,0x38,0x39,0x3A,0x3B,0x3C
                .db 0x41,0x42,0x43,0x44,0x45,0x46,0x47,0x48,0x49,0x4A,0x4B,0x4C
-------------------------------------------------------------------------------
```

```
;---------------BEGIN k48_getKey-----------------------------------------
;k48_getKey; waits until a key is pressed
;INPUT: none
;OUTPUT: r16 gets keycode
k48_getKey:
                push            r17
                push            r18
_k48_get2:
                rcall           k48_kbhit       ;use k16_kbhit to check for key
                                                ;r16 stores the right keycode
                cpi             r16, 0xFF       ;if no key is pressed then
                breq            _k48_get2       ;keep checking
                mov             r17,r16

_pressdeb:
                clr             r18
                do              0
                                rcall           k48_kbhit
                                cp              r16,r17
                                breq            _pressdeb
                                dec             r18
                whilene         0
_waitrelease:
                rcall           k48_kbhit
                cp              r16, r17
                breq            _pressdeb

_releasedeb:
                clr             r18
                do              0
                                rcall           k48_kbhit
                                cp              r16, r17
                                breq            _waitrelease

                                dec             r18
                whilene         0

                mov             r16, r17
                pop             r18
                pop             r17
                ret                             ;else return with keycode in r16

;---------------END k48_getKey-------------------------------------------


;---------------BEGIN k48_kbhit------------------------------------------
;k48_kbhit: return what key is pressed
;INPUT:         nothing
;OUTPUT:r16 = keycode or 0xFF if nothing is pressed


k48_kbhit:
                push  r17
                ;set the input/output of the 4 input pins and 12 output pins
                ldi             r16,0b00001111
                out             k48_DD1,r16
                ldi             r16,0b00000000
                out             k48_DD2,r16

                ;turn on the power line 1
                ldi             r16, 0b00000001
                out             k48_PORT1,r16
                nop
                in              r16,k48_PIN1
                in              r17,k48_PIN2
                rcall           checkline
                cpi             rLine, 0
                brne            founditrow1

                ;turn on the power line 2
                ldi             r16, 0b00000010
                out             k48_PORT1,r16
                nop
                in              r16, k48_PIN1
                in              r17,k48_PIN2
                rcall           checkline
                cpi             rLine, 0
                brne            founditrow2

                ;turn on the power line 3
                ldi             r16, 0b00000100
                out             k48_PORT1, r16
                nop
                in              r16,k48_PIN1
                in              r17,k48_PIN2
                rcall           checkline
                cpi             rLine, 0
                brne            founditrow3

                ;turn on the power line 4
                ldi             r16, 0b00001000
                out             k48_PORT1,r16
                nop
```

```
                in              r16,k48_PIN1
                in              r17,k48_PIN2
                rcall           checkline
                cpi             rLine, 0
                brne            founditrow4
                ;illegal
                ldi             r16, 0xFF
                pop             r17
                ret

founditrow1:
                ori             rLine,0b00010000
                mov             r16,rLine
                pop             r17
                ret
founditrow2:
                ori             rLine,0b00100000
                mov             r16,rLine
                pop             r17
                ret
founditrow3:
                ori             rLine,0b00110000
                mov             r16,rLine
                pop             r17
                ret
founditrow4:
                ori             rLine,0b01000000
                mov             r16,rLine
                pop             r17
                ret

;---------------BEGIN k48_kbhit-----------------------------------------------


;---------------BEGIN checkline subroutine----------------------------------
;r16 stores the upper 8 bits, r17 stores lower 8 bits
;rLine will store the correct line, 1-12, 0 if nothing

checkline:

                andi            r16,0b11110000
                cpi             r16,0b00000000
                breq            line8orlower
                cpi             r16,0b10000000
                breq            line12
                cpi             r16,0b01000000
                breq            line11
                cpi             r16,0b00100000
                breq            line10
                cpi             r16,0b00010000
                breq            line9

line8orlower:
                cpi             r17,0b00000000
                breq            line0
                cpi             r17,0b00000001
                breq            line1
                cpi             r17,0b00000010
                breq            line2
                cpi             r17,0b00000100
                breq            line3
                cpi             r17,0b00001000
                breq            line4
                cpi             r17,0b00010000
                breq            line5
                cpi             r17,0b00100000
                breq            line6
                cpi             r17,0b01000000
                breq            line7
                cpi             r17,0b10000000
                breq            line8

line0:
                ldi             rLine, 0         ;error
                ret
line1:
                ldi             rLine, 1
                ret
line2:
                ldi             rLine, 2
                ret
line3:
                ldi             rLine, 3
                ret
line4:
                ldi             rLine, 4
                ret
line5:
                ldi             rLine, 5
                ret
line6:
                ldi             rLine, 6
                ret
line7:
                ldi             rLine, 7
                ret
```

```
line8:
                ldi             rLine, 8
                ret
line9:
                ldi             rLine, 9
                ret
line10:
                ldi             rLine, 10
                ret
line11:
                ldi             rLine, 11
                ret
line12:
                ldi             rLine, 12
                ret

;--------------------------------END checkline-----------------------------------------
```

LCD CODE

The following is the code we used to interface with the LCD. It is a bug−fixed and cleaner version of the code originally taken off the web and given by Prof Land to the class.

```
;***************************************
;LCD (16x1) routines
;
;All labels and variables starts with prefix:
;               lcd16_
;
;INPUT:
;  constants to be defined:
;               lcd16_DD        ; for setting the Data−Direction (e.g. .equ lcd16_DD=DDRD)
;               lcd16_PORT      ; the port used for controlling the keyboard
;               lcd16_PIN       ; the pin used for controlling the keyboard
;               lcd16_rs        ; the LCD rs pin number (e.g. .equ lcd16_rs=PD6)
;               lcd16_rw        ; the LCD rw pin number (e.g. .equ lcd16_rw=PD5)
;               lcd16_en        ; the LCD en pin number (e.g. .equ lcd16_en=PD4)
;  registers to be defined:
;               rLCDcharcnt ;to hold the current char location
;  1. timer0 overflow interrupt must call lcd16_timer0 when LCD is used (say, implement
states)
;  2. after finish resetting, and sei, must call lcd16_init
;
;LCD Connection:
;LCDpin  Connection
;−−−−−−−−−−−−−−−−−−
;1              gnd
;2              +5V
;3              trimpot wiper (for contrast control)
;4              Pin 6 of lcd16_PORT
;5              Pin 5 of lcd16_PORT
;6              Pin 4 of lcd16_PORT
;7−10           No Connection
;11             Pin 0 of lcd16_PORT
;12             Pin 1 of lcd16_PORT
;13             Pin 2 of lcd16_PORT
;14             Pin 3 of lcd16_PORT
;

;Quick Reference
;−−−−−−−−−−−−−−−
;0. lcd16_init ; initializes LCD
;1. lcd16_clr  ; clears LCD screen
;2. lcd16_putc ; displays character in r16
;               INPUT: r16 has character to display
;               OUTPUT: LCD gets r16
;3. lcd16_printflash           ;displays null−terminated string in flash
;               INPUT: Z points to correct position in flash ROM
;               OUTPUT: LCD gets string
;               DESTROYS: r16, r0
;4. lcd16_printRAM             ;displays null−terminated string in RAM
;               INPUT: Z points to correct position in RAM
;               OUTPUT: LCD gets string
;               DESTROYS: r16, r0
;5. lcd16_sendcmd                        ;sends command in r16
;               INPUT: r16 is the command to send
;               OUTPUT: LCD gets command
;               DESTROYS: r16

;***************************************************
;bring up LCD  (part of Bruce Land's LCD Demo code)
```

```
lcd16_init:
               push           r16
               push           r17

               rcall   _lcdinit         ;Initialize LCD module
      rcall   lcd16_clr          ;Clear LCD screen
      clr     rLCDcharcnt         ;zero the character count

               pop            r17
               pop            r16
               ret
;-------------------------------------------------


;-----------------------------------------
;THIS SECTION IS BRUCE LAND'S LCD CODE
;VIRTUALLY UNMODIFIED!!!
;
;Clear entire LCD
lcd16_clr:
                              push           r16
                              clr            rLCDcharcnt
_lcdclr:       ldi     rTemp,1          ;Clear LCD command
               rcall   _lcdcmd
                              pop            r16
               ret

;================================================
;       Initialize LCD module
_lcdinit:
               ldi     r16,0          ;Setup port pins
        out     lcd16_PORT,r16      ;Pull all pins low
        ldi     r16,0xff       ;All pins are outputs
        out     lcd16_DD,r16
        ldi     r17,256                       ;Wait at least 15 mS at power up
        rcall   _lcddelay

;       LCD specs call for 3 repetitions as follows
        ldi     r16,3           ;Function set
        out     lcd16_PORT,r16      ;to 8-bit mode
        nop                      ;nop is data setup time
        sbi     lcd16_PORT,lcd16_en      ;Toggle enable line
        cbi     lcd16_PORT,lcd16_en

        ldi     r17,256       ;Wait at least 15 mS
        rcall   _lcddelay

        ldi     r16,3           ;Function set
        out     lcd16_PORT,r16
        nop
        sbi     lcd16_PORT,lcd16_en      ;Toggle enable line
        cbi     lcd16_PORT,lcd16_en

        ldi     r17,256       ;Wait at least 15 ms
        rcall   _lcddelay

        ldi     r16,3           ;Function set
        out     lcd16_PORT,r16
        nop
        sbi     lcd16_PORT,lcd16_en      ;Toggle enable line
        cbi     lcd16_PORT,lcd16_en

        ldi     r17,256       ;Wait at least 15 ms
        rcall   _lcddelay

        ldi     r16,2           ;Function set, 4 line interface
        out     lcd16_PORT,r16
        nop
;       rcall   strobe          ;Toggle enable line
        sbi     lcd16_PORT,lcd16_en      ;Toggle enable line
        cbi     lcd16_PORT,lcd16_en

        ldi     r16,0b11110000 ;Make 4 data lines inputs
        out     lcd16_DD,r16

;       Finally,
;       At this point, the normal 4 wire command routine can be used

        ldi     r16,0b00100000 ;Function set, 4 wire, 1 line, 5x7 font
        rcall   _lcdcmd

        ldi     r16,0b00001100 ;Display on, no cursor, no blink
        rcall   _lcdcmd

        ldi     r16,0b00000110 ;Address increment, no scrolling
        rcall   _lcdcmd
```

```
        ret

;=============================================
;       Wait for LCD to go unbusy
_lcdwait:
                ldi     r16,0xF0        ;Make 4 data lines inputs
        out     lcd16_DD,r16
        sbi     lcd16_PORT,lcd16_rw     ;Set r/w pin to read
        cbi     lcd16_PORT,lcd16_rs     ;Set register select to command
_lcd_waitloop:
        sbi     lcd16_PORT,lcd16_en      ;Toggle enable line
        cbi     lcd16_PORT,lcd16_en
        in      r16,lcd16_PIN    ;Read busy flag
        ;Read, and ignore lower nibble
        sbi     lcd16_PORT,lcd16_en      ;Toggle enable line
        cbi     lcd16_PORT,lcd16_en

        sbrc    r16,3       ;Loop until done
        rjmp    _lcd_waitloop
        ret

;=============================================
;       Send command in r16 to LCD
;DESTROYS: r16
lcd16_sendcmd:
_lcdcmd:push    r16             ;Save character
        rcall   _lcdwait         ;Wait for LCD to be ready
        ldi     r16,0xFF        ;Make all port D pins outputs
        out     lcd16_DD,r16
        pop     r16             ;Get character back
        push    r16             ;Save another copy
        swap    r16             ;Get upper nibble
        andi    r16,0x0F        ;Strip off upper bits
        out     lcd16_PORT,r16      ;Put on port
        nop                      ;wait for data setup time
        sbi     lcd16_PORT,lcd16_en      ;Toggle enable line
        cbi     lcd16_PORT,lcd16_en

        pop     r16             ;Recall character
        andi    r16,0x0F        ;Strip off upper bits
        out     lcd16_PORT,r16      ;Put on port
        nop
        sbi     lcd16_PORT,lcd16_en      ;Toggle enable line
        cbi     lcd16_PORT,lcd16_en

        ldi     r16,0xF0        ;Make 4 data lines inputs
        out     lcd16_DD,r16
        ret

;=============================================
;Send character data in r16 to LCD
lcd16_putc:
        cpi     rLCDcharcnt,8       ;addressing changes at char #8!
        brne    _lcdput_0           ;at char 8, fix it
                push            r16
        ldi     r16,0xC0        ;Set address to last 8 chars
        rcall   _lcdcmd
                pop             r16
_lcdput_0:
                inc             rLCDcharcnt
_lcdput:push    r16
                push    r16             ;Save character
                rcall   _lcdwait         ;Wait for LCD to be ready
                ldi     r16,0xFF        ;Make all port D pins outputs
                out     lcd16_DD,r16

                pop     r16             ;Get character back
                push    r16             ;Save another copy
                swap    r16             ;Get upper nibble
                andi    r16,0x0F        ;Strip off upper bits
                out     lcd16_PORT,r16      ;Put on port
                sbi     lcd16_PORT,lcd16_rs     ;Register select set for data
                nop
                sbi     lcd16_PORT,lcd16_en      ;Toggle enable line
                cbi     lcd16_PORT,lcd16_en

        pop     r16             ;Recall character
        andi    r16,0x0F        ;Strip off upper bits
        out     lcd16_PORT,r16      ;Put on port
        sbi     lcd16_PORT,lcd16_rs     ;Register select set for data
        nop
        sbi     lcd16_PORT,lcd16_en      ;Toggle enable line
        cbi     lcd16_PORT,lcd16_en
        ;cbi    lcd16_PORT,lcd16_rs     ;--

        ldi     r16,0xF0        ;Make 4 data lines inputs
```

```
        out     lcd16_DD,r16
                pop             r16
        ret

lcd16_printflash:
_nextc:  lpm                     ;Get next character from Flash
        tst     r0              ;See if at end of message
        breq    _lcdend1            ;If so, next message
        cpi     rLCDcharcnt,8       ;addressing changes at char #8!
        brne    _lcdwrtit           ;at char 8, fix it
        ldi     r16,0xC0       ;Set address to last 8 chars
        rcall   _lcdcmd
_lcdwrtit:
                mov     r16,r0          ;Send it to the LCD
        rcall   _lcdput
        adiw    ZL,1            ;Increment Z-pointer
        inc     rLCDcharcnt         ;keep track of chars on display
        rjmp    _nextc           ;Loop for more
_lcdend1:
                ret

lcd16_printRAM:
_nextRAM:                               ;Get next character from Flash
                ld              r0, Z
        tst     r0              ;See if at end of message
        breq    _lcdendRAM1             ;If so, next message
        cpi     rLCDcharcnt,8       ;addressing changes at char #8!
        brne    _lcdwrtRAM           ;at char 8, fix it
        ldi     r16,0xC0       ;Set address to last 8 chars
        rcall   _lcdcmd
_lcdwrtRAM:
                mov     r16,r0          ;Send it to the LCD
        rcall   _lcdput
        adiw    ZL,1            ;Increment Z-pointer
        inc     rLCDcharcnt         ;keep track of chars on display
        rjmp    _nextRAM           ;Loop for more
_lcdendRAM1:   ret

;***** Timer 0 overflow interrupt handler
lcd16_timer0:
                push            r16
                set                      ;Set T flag
        ldi     r16,0                   ;Timer 0 off
        out     TCCR0,r16       ;Stop timer
                pop             r16
        reti                    ;Done, return

;***** Delay n*64 microseconds using timer 0, delay time passed in timeout
;r17 has the delay...
_lcddelay:
                push            r16
                in      r16,SREG        ;Save status register
        push    r16
                in              r16,TIMSK       ;Save the TIMSK
                push            r16
                ldi             r16,0x02        ;setup timer0 overflow interrupt
                out             TIMSK,r16       ;output to TIMSK
        out     TCNT0,r17
        clt                     ;Clear T
        ldi     r16,4                   ;Timer 0 prescaler, CK / 256
        out     TCCR0,r16       ;Run timer
dwait:  brtc    dwait           ;Wait for timer 0 interrupt to set T
                pop             r16                             ;Restore TIMSK
                out             TIMSK,r16
        pop     r16             ;Restore status register
        out     SREG,r16
                pop             r16
        ret
```

EEPROM Stub Code and Dictonary Data

As previously mentioned, we were unable to get our real EEPROM to work. As such, we are submitting the project with the EEPROM stub code we used for testing the project while we worked on the EEPROM. It uses the Flash ROM and the functions are used in the same way as the real EEPROM code.

```
;*********************EEPROM STUB**********************
e256k_getc:
                push            ZL
                push            ZH
```

```
                push        r0

                mov         ZL, rE256k_ADD0
                mov         ZH, rE256k_ADD1
                lpm
                adiw        ZL, 1
                mov         rE256k_ADD0, ZL
                mov         rE256k_ADD1, ZH
                mov         r16, r0

                pop         r0
                pop         ZH
                pop         ZL
                ret
;*****************************************************
;
FakeEEPROM:
.include "ee476.dic"
```

```
                    .db             0x0A, 'A', 'B', 'A', 'D', 0, 'c', 'e', 'n', 't', 'u', 'r', 'y'
                    .db             0x0A, 'A', 'B', 'A', 'D', 'I', 0, 'e', 't', 'e', 'r', 'n', 'a', 'l', 'i', 'm',
'm', 'o', 'r', 't', 'a', 'l'
                    .db             0x0A, 'A', 'B', 'A', 'H', 0, 'f', 'a', 't', 'h', 'e', 'r'
                    .db             0x0A, 'A', 'B', 'A', 'I', 0, 'n', 'e', 'g', 'l', 'e', 'c', 't'
                    .db             0x0A, 'A', 'B', 'A', 'N', 'G', 0, 'e', 'l', 'd', 'e', 'r', ' ', 'b', 'r', 'o',
't', 'h', 'e', 'r'
                    .db             0x0A, 'A', 'B', 'D', 'I', 0, 's', 'l', 'a', 'v', 'e'
                    .db             0x0A, 'A', 'B', 'J', 'A', 'D', 0, 'a', 'l', 'p', 'h', 'a', 'b', 'e', 't'
                    .db             0x0A, 'A', 'B', 'U', 0, 'a', 's', 'h'
                    .db             0x0A, 'A', 'D', 'A', 0, 'h', 'a', 'v', 'e', 'p', 'o', 's', 's', 'e', 's', 's',
'c', 'o', 'n', 't', 'a', 'i', 'n'
                    .db             0x0A, 'A', 'D', 'A', 'K', 'A', 'L', 'A', 0, 's', 'o', 'm', 'e', 't', 'i', 'm',
'e', 's'
                    .db             0x0A, 'A', 'D', 'A', 'K', 'A', 'L', 'A', 'N', 'Y', 'A', 0, 's', 'o', 'm', 'e',
't', 'i', 'm', 'e', 's'
                    .db             0x0A, 'A', 'D', 'A', 'L', 'A', 'H', 0, 'i', 's', 'a', 'r', 'e', 'a', 'm', 'w',
'a', 's', 'w', 'e', 'r', 'e'
                    .db             0x0A, 'A', 'D', 'A', 'P', 'U', 'N', 0, 'w', 'i', 't', 'h', ' ', 'r', 'e', 'g',
'a', 'r', 'd', ' ', 't', 'o'
                    .db             0x0A, 'B', 'E', 'R', 'A', 'D', 'A', 0, '(', '1', ')', ' ', 'r', 'i', 'c', 'h',
' ', '(', '2', ')', ' ', 't', 'o', ' ', 'b', 'e'
                    .db             0x0A, 'D', 'I', 'A', 'D', 'A', 'K', 'A', 'N', 0, 't', 'o', ' ', 'b', 'e', ' ',
'h', 'e', 'l', 'd'
                    .db             0x0A, 'A', 'D', 'A', 'T', 0, 'c', 'u', 's', 't', 'o', 'm', 's', 'o', 'c', 'i',
'a', 'l', ' ', 't', 'r', 'a', 'd', 'i', 't', 'i', 'o', 'n', ' ', 'o', 'r', ' ', 'p', 'r', 'a', 'c', 't', 'i', 'c', 'e'
                    .db             0x0A, 'A', 'D', 'U', 0, '(', '1', ')', 'c', 'o', 'm', 'p', 'l', 'a', 'i', 'n',
' ', '(', '2', ')', 'c', 'o', 'm', 'p', 'e', 't', 'e'
                    .db             0x0A, 'B', 'E', 'R', 'A', 'D', 'U', 0, '(', '1', ')', 't', 'o', ' ', 'r', 'e',
'p', 'o', 'r', 't', ' ', '(', '2', ')', 't', 'o', ' ', 'c', 'o', 'm', 'p', 'e', 't', 'e'
                    .db             0x0A, 'A', 'D', 'U', 'A', 'N', 0, 'c', 'o', 'm', 'p', 'l', 'a', 'i', 'n', 't'
                    .db             0x0A, 'A', 'G', 'A', 'H', 0, 'm', 'u', 'r', 'm', 'u', 'r', 'i', 'n', 'g', ' ',
's', 'o', 'u', 'n', 'd'
                    .db             0x0A, 'A', 'G', 'A', 'K', 0, 'g', 'u', 'e', 's', 's'
                    .db             0x0A, 'A', 'G', 'A', 'K', 'N', 'Y', 'A', 0, 'p', 'e', 'r', 'h', 'a', 'p', 's',
'm', 'a', 'y', 'b', 'e'
                    .db             0x0A, 'B', 'E', 'R', 'A', 'G', 'A', 'K', 0, 'n', 'o', 't', ' ', 's', 'u', 'r',
'e'
                    .db             0x0A, 'M', 'E', 'N', 'G', 'A', 'G', 'A', 'K', 0, 't', 'o', ' ', 'g', 'u', 'e',
's', 's', 't', 'o', ' ', 'e', 's', 't', 'i', 'm', 'a', 't', 'e'
                    .db             0x0A, 'A', 'G', 'A', 'M', 0, 'i', 'm', 'm', 'e', 'a', 's', 'u', 'r', 'a', 'b',
'l', 'e', 'i', 'm', 'm', 'e', 'n', 's', 'e'
                    .db             0x0A, 'A', 'G', 'A', 'M', 'A', 0, 'r', 'e', 'l', 'i', 'g', 'i', 'o', 'n'
                    .db             0x0A, 'A', 'G', 'A', 'R', 0, 's', 'o', ' ', 't', 'h', 'a', 't'
                    .db             0x0A, 'A', 'G', 'A', 'S', 0, 'g', 'n', 'a', 't', 's', 'a', 'n', 'd', 'f', 'l',
'y'
                    .db             0x0A, 'A', 'G', 'I', 'H', 0, 'd', 'i', 's', 't', 'r', 'i', 'b', 'u', 't', 'e'
                    .db             0x0A, 'A', 'G', 'U', 'K', 0, 'p', 'e', 'n', 'd', 'a', 'n', 't'
                    .db             0x0A, 'A', 'G', 'U', 'N', 'G', 0, 's', 'u', 'p', 'r', 'e', 'm', 'e'
                    .db             0x0A, 'K', 'E', 'A', 'G', 'U', 'N', 'G', 'A', 'N', 0, 's', 'u', 'p', 'r', 'e',
'm', 'a', 'c', 'y'
                    .db             0x0A, 'M', 'E', 'N', 'G', 'A', 'G', 'U', 'N', 'G', 'K', 'A', 'N', 0, 't', 'o',
' ', 'h', 'o', 'n', 'o', 'u', 'r', 't', 'o', ' ', 'r', 'e', 'v', 'e', 'r', 'e'
                    .db             0x0A, 'A', 'H', 0, 'e', 'x', 'c', 'l', 'a', 'm', 'a', 't', 'i', 'o', 'n', ' ',
'o', 'f', ' ', 's', 'u', 'r', 'p', 'r', 'i', 's', 'e'
                    .db             0x0A, 'A', 'H', 'A', 0, 'e', 'x', 'c', 'l', 'a', 'm', 'a', 't', 'i', 'o', 'n',
' ', 'o', 'f', ' ', 't', 'r', 'i', 'u', 'm', 'p', 'h'
                    .db             0x0A, 'A', 'H', 'A', 'D', 0, 'S', 'u', 'n', 'd', 'a', 'y'
                    .db             0x0A, 'A', 'H', 'L', 'I', 0, '(', '1', ')', 'm', 'e', 'm', 'b', 'e', 'r', ' ',
'(', '2', ')', 'e', 'x', 'p', 'e', 'r', 't'
                    .db             0x0A, 'A', 'H', 'M', 'A', 'K', 0, 's', 't', 'u', 'p', 'i', 'd'
                    .db             0x0A, 'A', 'I', 'R', 0, 'w', 'a', 't', 'e', 'r'
                    .db             0x0A, 'A', 'I', 'S', 0, 'i', 'c', 'e'
                    .db             0x0A, 'A', 'J', 'A', 'I', 'B', 0, 'b', 'a', 'f', 'f', 'l', 'i', 'n', 'g', 'i',
'n', 's', 'c', 'r', 'u', 't', 'a', 'b', 'l', 'e', 'm', 'i', 'r', 'a', 'c', 'u', 'l', 'o', 'u', 's'
                    .db             0x0A, 'A', 'J', 'A', 'K', 0, 'i', 'n', 'v', 'i', 't', 'e'
                    .db             0x0A, 'A', 'J', 'A', 'L', 0, 'd', 'o', 'o', 'm', 'd', 'e', 'a', 't', 'h'
                    .db             0x0A, 'A', 'J', 'A', 'R', 0, 't', 'e', 'a', 'c', 'h'
                    .db             0x0A, 'K', 'U', 'R', 'A', 'N', 'G', ' ', 'A', 'J', 'A', 'R', 0, 'd', 'i', 's',
'r', 'e', 's', 'p', 'e', 'c', 't', 'f', 'u', 'l'
                    .db             0x0A, 'A', 'J', 'A', 'R', 'A', 'N', 0, 't', 'e', 'a', 'c', 'h', 'i', 'n', 'g',
                    .db             0x0A, 'B', 'E', 'L', 'A', 'J', 'A', 'R', 0, 't', 'o', ' ', 'l', 'e', 'a', 'r',
'n'
                    .db             0x0A, 'B', 'E', 'R', 'A', 'J', 'A', 'R', 0, 'e', 'd', 'u', 'c', 'a', 't', 'e',
'd', 'c', 'o', 'u', 'r', 't', 'e', 'o', 'u', 's'
                    .db             0x0A, 'M', 'E', 'M', 'P', 'E', 'L', 'A', 'J', 'A', 'R', 'I', 0, 't', 'o', ' ',
'l', 'e', 'a', 'r', 'n', ' ', 'i', 'n', ' ', 'g', 'r', 'e', 'a', 't', 'e', 'r', ' ', 'd', 'e', 't', 'a', 'i', 'l'
                    .db             0x0A, 'M', 'E', 'N', 'G', 'A', 'J', 'A', 'R', 0, 't', 'o', ' ', 't', 'e', 'a',
```

```
'c', 'h'
                .db                 0x0A, 'M', 'E', 'N', 'G', 'A', 'J', 'A', 'R', 'K', 'A', 'N', 0, 't', 'o', ' ',
't', 'e', 'a', 'c', 'h'
                .db                 0x0A, 'P', 'E', 'L', 'A', 'J', 'A', 'R', 0, 's', 't', 'u', 'd', 'e', 'n', 't'
                .db                 0x0A, 'A', 'J', 'U', 'K', 0, 'i', 'm', 'i', 't', 'a', 't', 'e', ' ', 'm', 'o',
'c', 'k', 'i', 'n', 'g', 'l', 'y', 't', 'e', 'a', 's', 'e'
                .db                 0x0A, 'M', 'E', 'N', 'G', 'A', 'J', 'U', 'K', 0, 't', 'o', ' ', 'i', 'm', 'i',
't', 'a', 't', 'e', ' ', 'm', 'o', 'c', 'k', 'i', 'n', 'g', 'l', 'y', 't', 'o', ' ', 't', 'e', 'a', 's', 'e'
                .db                 0x0A, 'A', 'J', 'U', 'K', 'A', 'N', 0, 'm', 'o', 'c', 'k', 'e', 'r', 'y', 'r',
'i', 'd', 'i', 'c', 'u', 'l', 'e', 't', 'e', 'a', 's', 'i', 'n', 'g'
                .db                 0x0A, 'A', 'K', 'A', 'D', 0, 'c', 'o', 'n', 't', 'r', 'a', 'c', 't', 'p', 'r',
'o', 'm', 'i', 's', 'e'
                .db                 0x0A, 'A', 'K', 'A', 'I', 'D', 0, 'r', 'e', 'l', 'i', 'g', 'i', 'o', 'u', 's',
' ', 'd', 'o', 'g', 'm', 'a'
                .db                 0x0A, 'A', 'K', 'A', 'L', 0, 'a', 'b', 'i', 'l', 'i', 't', 'y', ' ', 't', 'o',
' ', 't', 'h', 'i', 'n', 'k', ' ', 'a', 'n', 'd', ' ', 'r', 'e', 'a', 's', 'o', 'n'
                .db                 0x0A, 'A', 'K', 'A', 'L', ' ', 'B', 'U', 'D', 'I', 0, 'c', 'o', 'm', 'm', 'o',
'n', ' ', 's', 'e', 'n', 's', 'e'
                .db                 0x0A, 'A', 'K', 'A', 'L', '-', 'A', 'K', 'A', 'L', 0, 't', 'o', ' ', 'p', 'r',
'e', 't', 'e', 'n', 'd', ' ', 'o', 'r', ' ', 'f', 'e', 'i', 'g', 'n'
                .db                 0x0A, 'A', 'K', 'H', 'L', 'A', 'K', 0, 'b', 'e', 'h', 'a', 'v', 'i', 'o', 'u',
'r', 'c', 'h', 'a', 'r', 'a', 'c', 't', 'e', 'r', 'm', 'o', 'r', 'a', 'l', 's'
                .db                 0x0A, 'A', 'K', 'I', 'B', 'A', 'T', 0, 'c', 'o', 'n', 's', 'e', 'q', 'u', 'e',
'n', 'c', 'e', 'r', 'e', 's', 'u', 'l', 't'
                .db                 0x0A, 'A', 'K', 'I', 'B', 'A', 'T', 'N', 'Y', 'A', 0, 'a', 's', ' ', 'a', ' ',
'r', 'e', 's', 'u', 'l', 't'
                .db                 0x0A, 'B', 'E', 'R', 'A', 'K', 'I', 'B', 'A', 'T', 0, 'r', 'e', 's', 'u', 'l',
't', 'e', 'd', ' ', 'i', 'n'
                .db                 0x0A, 'M', 'E', 'N', 'G', 'A', 'K', 'I', 'B', 'A', 'T', 'K', 'A', 'N', 0, 't',
'o', ' ', 'c', 'a', 'u', 's', 'e', 't', 'o', ' ', 'r', 'e', 's', 'u', 'l', 't', ' ', 'i', 'n'
                .db                 0x0A, 'A', 'K', 'I', 'D', 'A', 'H', 0, 'b', 'e', 'l', 'i', 'e', 'f', 'f', 'a',
'i', 't', 'h', 'c', 'o', 'n', 'f', 'i', 'd', 'e', 'n', 'c', 'e'
                .db                 0x0A, 'A', 'K', 'I', 'K', 'A', 'H', 0, 'h', 'e', 'a', 'd', '-', 's', 'h', 'a',
'v', 'i', 'n', 'g', ' ', 'o', 'f', ' ', 'a', 'n', ' ', 'i', 'n', 'f', 'a', 'n', 't'
                .db                 0x0A, 'A', 'K', 'I', 'L', 0, 'c', 'l', 'e', 'v', 'e', 'r'
                .db                 0x0A, 'A', 'K', 'I', 'L', ' ', 'B', 'A', 'L', 'I', 'G', 'H', 0, 'g', 'r', 'o',
'w', 'n', ' ', 'u', 'p', 'a', 'd', 'o', 'l', 'e', 's', 'c', 'e', 'n', 't'
                .db                 0x0A, 'A', 'K', 'R', 'A', 'B', 0, 'i', 'n', 't', 'i', 'm', 'a', 't', 'e', ' ',
'i', 'n', ' ', 'r', 'e', 'l', 'a', 't', 'i', 'o', 'n', 's', 'h', 'i', 'p'
                .db                 0x0A, 'A', 'K', 'T', 'A', 0, 'a', 'c', 't', 'l', 'a', 'w', ' ', 'p', 'a', 's',
's', 'e', 'd', ' ', 'b', 'y', ' ', 'l', 'e', 'g', 'i', 's', 'l', 'a', 't', 'i', 'v', 'e', ' ', 'b', 'o', 'd', 'y'
                .db                 0x0A, 'A', 'K', 'U', 0, 'm', 'e', 'I'
                .db                 0x0A, 'A', 'K', 'U', 'A', 'N', 0, 'c', 'o', 'n', 'f', 'e', 's', 's', 'i', 'o',
'n', 'a', 'c', 'c', 'e', 'p', 't', 'a', 'n', 'c', 'e'
                .db                 0x0A, 'M', 'E', 'M', 'P', 'E', 'R', 'A', 'K', 'U', 'I', 0, 't', 'o', ' ', 'c',
'e', 'r', 't', 'i', 'f', 'y'
                .db                 0x0A, 'M', 'E', 'M', 'P', 'E', 'R', 'A', 'K', 'U', 'K', 'A', 'N', 0, 't', 'o',
' ', 'c', 'e', 'r', 't', 'i', 'f', 'y', 't', 'o', ' ', 'r', 'e', 'c', 'o', 'm', 'm', 'e', 'n', 'd'
                .db                 0x0A, 'M', 'E', 'N', 'G', 'A', 'K', 'U', 0, 't', 'o', ' ', 'c', 'o', 'n', 'f',
'e', 's', 's', 't', 'o', ' ', 'a', 'd', 'm', 'i', 't'
                .db                 0x0A, 'M', 'E', 'N', 'G', 'A', 'K', 'U', 'I', 0, 't', 'o', ' ', 'a', 'd', 'm',
'i', 't', ' ', 't', 'o', ' ', 'a', 'c', 'c', 'e', 'p', 't', 't', 'o', ' ', 'a', 'c', 'k', 'n', 'o', 'w', 'l', 'e', 'd',
'g', 'e'
                .db                 0x0A, 'A', 'N', 'C', 'A', 'M', 0, 'm', 'e', 'n', 'a', 'c', 'e', 't', 'h', 'r',
'e', 'a', 't', 'e', 'n'
                .db                 0x0A, 'A', 'N', 'D', 'A', 0, 'y', 'o', 'u'
                .db                 0x0A, 'A', 'N', 'D', 'A', 'I', 0, 'i', 'f', 's', 'u', 'p', 'p', 'o', 's', 'e'
                .db                 0x0A, 'A', 'N', 'D', 'A', 'I', 'N', 'Y', 'A', 0, 's', 'u', 'p', 'p', 'o', 's',
'e'
                .db                 0x0A, 'A', 'N', 'D', 'A', 'I', 'A', 'N', 0, 'a', 's', 's', 'u', 'm', 'p', 't',
'i', 'o', 'n'
                .db                 0x0A, 'B', 'E', 'R', 'A', 'N', 'D', 'A', 'I', 0, 't', 'o', ' ', 'd', 'i', 's',
'c', 'u', 's', 's', 't', 'o', ' ', 'd', 'e', 'b', 'a', 't', 'e'
                .db                 0x0A, 'A', 'N', 'D', 'A', 'M', 0, 'b', 'e', 'a', 'u', 't', 'i', 'c', 'i', 'a',
'n', ' ', 'w', 'h', 'o', ' ', 'd', 'o', 'e', 's', ' ', 't', 'h', 'e', ' ', 'm', 'a', 'k', 'e', '-', 'u', 'p', ' ', 'f',
'o', 'r', ' ', 't', 'h', 'e', ' ', 'b', 'r', 'i', 'd', 'e', ' ', 'a', 'n', 'd'
                .db                 0x0A, 'B', 'R', 'I', 'D', 'E', 'G', 'R', 'O', 'O', 'M', 0
                .db                 0x0A, 'A', 'N', 'D', 'A', 'S', 0, 'a', 'n', 'v', 'i', 'l', 'c', 'h', 'o', 'p',
'p', 'i', 'n', 'g', ' ', 'b', 'l', 'o', 'c', 'k'
                .db                 0x0A, 'A', 'N', 'D', 'U', 'H', 0, 's', 'u', 's', 'p', 'e', 'n', 'd', ' ', 'b',
'y', ' ', 'm', 'e', 'a', 'n', 's', ' ', 'o', 'f', ' ', 'a', ' ', 's', 'l', 'i', 'n', 'g'
                .db                 0x0A, 'A', 'N', 'E', 'H', 0, 'b', 'i', 'z', 'a', 'r', 'r', 'e', 'o', 'd', 'd',
'u', 'n', 'c', 'o', 'n', 'v', 'e', 'n', 't', 'i', 'o', 'n', 'a', 'l', 'f', 'u', 'n', 'n', 'y', 's', 't', 'r', 'a', 'n',
'g', 'e'
                .db                 0x0A, 'A', 'N', 'E', 'K', 'A', 0, 'm', 'u', 'l', 't', 'i', 'f', 'a', 'r', 'i',
'o', 'u', 's', 'v', 'a', 'r', 'i', 'e', 'd', 'a', 'l', 'l', ' ', 's', 'o', 'r', 't', 's', ' ', 'o', 'f'
                .db                 0x0A, 'A', 'N', 'G', 'A', 'N', 0, 'i', 'm', 'a', 'g', 'i', 'n', 'a', 't', 'i',
'o', 'n'
                .db                 0x0A, 'A', 'N', 'G', 'A', 'N', '-', 'A', 'N', 'G', 'A', 'N', 0, 'i', 'm', 'a',
'g', 'i', 'n', 'a', 't', 'i', 'o', 'n', 'a', ' ', 'd', 'a', 'y', '-', 'd', 'r', 'e', 'a', 'm'
                .db                 0x0A, 'B', 'E', 'R', 'A', 'N', 'G', 'A', 'N', '-', 'A', 'N', 'G', 'A', 'N', 0,
't', 'o', ' ', 'd', 'a', 'y', '-', 'd', 'r', 'e', 'a', 'm'
                .db                 0x0A, 'M', 'E', 'N', 'G', 'A', 'N', 'G', 'A', 'N', '-', 'A', 'N', 'G', 'A',
'N', 'K', 'A', 'N', 0, 't', 'o', ' ', 'w', 'i', 's', 'h', ' ', 'f', 'o', 'r', 't', 'o', ' ', 'd', 'a', 'y', '-', 'd',
'r', 'e', 'a', 'm', ' ', 'a', 'b', 'o', 'u', 't'
                .db                 0x0A, 'A', 'N', 'G', 'A', 'U', 0, 'l', 'o', 'v', 'e', 's', 'i', 'c', 'k'
                .db                 0x0A, 'M', 'E', 'M', 'P', 'E', 'R', 'B', 'A', 'D', 'A', 'N', 'K', 'A', 'N', 0,
't', 'o', ' ', 'c', 'o', 'r', 'p', 'o', 'r', 'a', 't', 'i', 'z', 'e'
                .db                 0x0A, 'P', 'E', 'R', 'B', 'A', 'D', 'A', 'N', 'A', 'N', 0, 'c', 'o', 'r', 'p',
'o', 'r', 'a', 't', 'i', 'o', 'n', 'i', 'n', 's', 't', 'i', 't', 'u', 't', 'i', 'o', 'n'
                .db                 0x0A, 'B', 'A', 'D', 'I', 'K', 0, 'd', 'a', 'g', 'g', 'e', 'r'
                .db                 0x0A, 'B', 'A', 'D', 'U', 'T', 0, 'c', 'l', 'o', 'w', 'n', 'c', 'o', 'm', 'e',
'd', 'i', 'a', 'n'
                .db                 0x0A, 'B', 'A', 'D', 'U', 'I', 0, 'B', 'e', 'd', 'o', 'u', 'i', 'n'
                .db                 0x0A, 'B', 'A', 'G', 'A', 'I', 0, 'a', 's', ' ', ' ', 'i', 'f', 'l', 'i', 'k', 'e'
                .db                 0x0A, 'B', 'A', 'G', 'A', 'I', 'M', 'A', 'N', 'A', 0, 'h', 'o', 'w', 'e', 'v',
'e', 'r', 'n', 'o', ' ', 'm', 'a', 't', 't', 'e', 'r', ' ', 'h', 'o', 'w', ' ', 'o', 'r', ' ', 'w', 'h', 'a', 't'
                .db                 0x0A, 'S', 'E', 'B', 'A', 'G', 'A', 'I', 'M', 'A', 'N', 'A', 0, 'a', 's', 'l',
'i', 'k', 'e'
                .db                 0x0A, 'B', 'A', 'G', 'A', 'S', 'I', 0, 'b', 'a', 'g', 'g', 'a', 'g', 'e', 'c',
'a', 'r', 'g', 'o'
                .db                 0x0A, 'B', 'A', 'G', 'H', 'A', 'L', 0, 'm', 'u', 'l', 'e'
                .db                 0x0A, 'B', 'A', 'G', 'I', 0, '(', '1', ')', 'f', 'o', 'r', ' ', '(', '2', ')',
'g', 'i', 'v', 'e'
                .db                 0x0A, 'B', 'A', 'G', 'I', 'N', 'D', 'A', 0, 'H', 'i', 's', ' ', 'o', 'r', ' ',
'H', 'e', 'r', ' ', 'M', 'a', 'j', 'e', 's', 't', 'y'
                .db                 0x0A, 'B', 'A', 'G', 'U', 'R', 0, 'l', 'a', 'r', 'g', 'e', ' ', 'a', 'n', 'd',
' ', 'c', 'l', 'u', 'm', 's', 'y'
```

```
                    .db                 0x0A, 'B', 'A', 'G', 'U', 'S', 0, 'n', 'i', 'c', 'e', 'p', 'l', 'e', 'a', 's',
'a', 'n', 't', 'g', 'o', 'o', 'd'
                    .db                 0x0A, 'B', 'A', 'H', 0, 'd', 'e', 'l', 'u', 'g', 'e', 'f', 'l', 'o', 'o', 'd'
                    .db                 0x0A, 'B', 'A', 'H', 'A', 'G', 'I', 0, 'a', 'l', 'l', 'o', 't', 'd', 'i', 's',
't', 'r', 'i', 'b', 'u', 't', 'e', 'd', 'i', 'v', 'i', 'd', 'e'
                    .db                 0x0A, 'B', 'A', 'H', 'A', 'R', 'I', 0, 'a', 'n', 'c', 'i', 'e', 'n', 't'
                    .db                 0x0A, 'B', 'A', 'H', 'A', 'R', 'U', 0, 'n', 'e', 'w'
                    .db                 0x0A, 'B', 'A', 'H', 'U', 0, 's', 'h', 'o', 'u', 'l', 'd', 'e', 'r'
                    .db                 0x0A, 'B', 'A', 'I', 'D', 'U', 'R', 'I', 0, 'o', 'p', 'a', 'l'
                    .db                 0x0A, 'K', 'E', 'B', 'A', 'K', 'A', 'R', 'A', 'N', 0, 'a', ' ', 'f', 'i', 'r',
'e'
                    .db                 0x0A, 'M', 'E', 'M', 'B', 'A', 'K', 'A', 'R', 0, 't', 'o', ' ', 'b', 'u', 'r',
'n'
                    .db                 0x0A, 'T', 'E', 'R', 'B', 'A', 'K', 'A', 'R', 0, 'o', 'n', ' ', 'f', 'i', 'r',
'e', 'b', 'u', 'r', 'n', 't'
                    .db                 0x0A, 'B', 'A', 'K', 'A', 'T', 0, 't', 'a', 'l', 'e', 'n', 't', 'f', 'l', 'a',
'i', 'r', 'a', 'p', 't', 'i', 't', 'u', 'd', 'e'
                    .db                 0x0A, 'B', 'A', 'K', 'A', 'U', 0, 'm', 'a', 'n', 'g', 'r', 'o', 'v', 'e'
                    .db                 0x0A, 'B', 'A', 'K', 'A', 'W', 'A', 'L', 'I', 0, 'a', ' ', 'm', 'e', 'd', 'i',
'c', 'i', 'n', 'a', 'l', ' ', 'h', 'e', 'r', 'b', 'a', 'l', ' ', 'p', 'l', 'a', 'n', 't'
                    .db                 0x0A, 'B', 'A', 'K', 'H', 'I', 'L', 0, 's', 't', 'i', 'n', 'g', 'y', 'm', 'i',
's', 'e', 'r', 'l', 'y'
                    .db                 0x0A, 'B', 'A', 'K', 'I', 0, 'b', 'a', 'l', 'a', 'n', 'c', 'e', 'r', 'e', 'm',
'a', 'i', 'n', 'd', 'e', 'r'
                    .db                 0x0A, 'B', 'A', 'K', 'T', 'I', 0, 'd', 'e', 'v', 'o', 't', 'i', 'o', 'n', 'd',
'e', 'd', 'i', 'c', 'a', 't', 'i', 'o', 'n'
                    .db                 0x0A, 'B', 'A', 'L', 'A', 'S', 0, 'r', 'e', 'c', 'i', 'p', 'r', 'o', 'c', 'a',
't', 'e', 'r', 'e', 'p', 'l', 'y', 'a', 'v', 'e', 'n', 'g', 'e'
                    .db                 0x0A, 'M', 'E', 'M', 'B', 'A', 'L', 'A', 'S', 0, 't', 'o', ' ', 'r', 'e', 'c',
'i', 'p', 'r', 'o', 'c', 'a', 't', 'e'
                    .db                 0x0A, 'K', 'A', 'C', 'A', 'U', 0, '(', '1', ')', 't', 'o', ' ', 'h', 'a', 'r',
'a', 's', 's', 't', 'o', ' ', 'd', 'i', 's', 't', 'u', 'r', 'b', ' ', '(', '2', ')', 't', 'o', ' ', 's', 't', 'i', 'r'
                    .db                 0x0A, 'K', 'A', 'C', 'I', 'P', 0, 'n', 'u', 't', 'c', 'r', 'a', 'c', 'k', 'e',
'r'
                    .db                 0x0A, 'K', 'A', 'C', 'U', 'K', 0, 'c', 'r', 'o', 's', 's', '-', 'b', 'r', 'e',
'd'
                    .db                 0x0A, 'K', 'A', 'C', 'U', 'K', 'A', 'N', 0, 'o', 'f', 'f', 's', 'p', 'r', 'i',
'n', 'g', ' ', 'o', 'f', ' ', 'c', 'r', 'o', 's', 's', '-', 'b', 'r', 'e', 'e', 'd'
                    .db                 0x0A, 'K', 'A', 'C', 'U', 'N', 'G', 0, 'm', 'a', 'n', 't', 'i', 's'
                    .db                 0x0A, 'K', 'A', 'D', 'A', 'M', 0, 's', 'o', 'l', 'e', ' ', 'o', 'f', ' ', 'f',
'o', 'o', 't'
                    .db                 0x0A, 'K', 'A', 'D', 'A', 'N', 'G', '-', 'K', 'A', 'D', 'A', 'N', 'G', 0, 's',
'o', 'm', 'e', 't', 'i', 'm', 'e', 's'
                    .db                 0x0A, 'K', 'A', 'D', 'A', 'N', 'G', ' ', 'K', 'A', 'L', 'A', 0, 's', 'o', 'm',
'e', 't', 'i', 'm', 'e', 's', 'n', 'o', 'w', ' ', 'a', 'n', 'd', ' ', 't', 'h', 'e', 'n'
                    .db                 0x0A, 'K', 'A', 'D', 'A', 'R', 0, 'r', 'a', 't', 'e', 'r', 'a', 't', 'i', 'o',
'p', 'r', 'o', 'p', 'o', 'r', 't', 'i', 'o', 'n'
                    .db                 0x0A, 'K', 'A', 'D', 'I', 0, 'M', 'u', 's', 'l', 'i', 'm', ' ', 'j', 'u', 'd',
'g', 'e'
                    .db                 0x0A, 'K', 'A', 'D', 'U', 'K', 0, 'f', 'o', 'o', 'l', ' ', '(', 'o', 'f', ' ',
'M', 'a', 'l', 'a', 'y', ' ', 'f', 'o', 'l', 'k', 'l', 'o', 'r', 'e', ')'
                    .db                 0x0A, 'K', 'O', 'C', 'E', 'K', 0, 'p', 'o', 'c', 'k', 'e', 't'
                    .db                 0x0A, 'K', 'O', 'C', 'O', 'K', 0, 't', 'o', ' ', 's', 'h', 'a', 'k', 'e', 't',
'o', ' ', 'm', 'i', 'x', 't', 'o', ' ', 'i', 'n', 'c', 'i', 't', 'e'
                    .db                 0x0A, 'K', 'O', 'D', 'I', 0, 'a', ' ', 's', 'c', 'o', 'r', 'e', ' ', '(', 't',
'w', 'e', 'n', 't', 'y', ')'
                    .db                 0x0A, 'K', 'O', 'D', 'O', 'K', 0, 'f', 'r', 'o', 'g'
                    .db                 0x0A, 'K', 'O', 'H', 'O', 'N', 'G', 0, 'f', 'e', 't', 'i', 'd', 's', 't', 'i',
'n', 'k', 'i', 'n', 'g'
                    .db                 0x0A, 'K', 'O', 'K', 'O', 'K', 0, 't', 'o', ' ', 'c', 'r', 'o', 'w'
                    .db                 0x0A, 'K', 'O', 'L', 'A', 'H', 0, 's', 'q', 'u', 'a', 'r', 'e', ' ', 's', 't',
'o', 'n', 'e', ' ', 'o', 'r', ' ', 'c', 'e', 'm', 'e', 'n', 't', ' ', 'w', 'a', 't', 'e', 'r', ' ', 't', 'a', 'n', 'k'
                    .db                 0x0A, 'K', 'O', 'L', 'A', 'M', 0, 'p', 'o', 'n', 'd'
                    .db                 0x0A, 'K', 'O', 'L', 'E', 0, 'm', 'u', 'g'
                    .db                 0x0A, 'K', 'O', 'L', 'E', 'K', 0, 'c', 'a', 'n', 'o', 'e'
                    .db                 0x0A, 'K', 'O', 'L', 'O', 'T', 0, 'o', 'l', 'd', '-', 'f', 'a', 's', 'h', 'i',
'o', 'n', 'e', 'd', 'o', 'u', 't', '-', 'o', 'f', '-', 'd', 'a', 't', 'e'
                    .db                 0x0A, 'K', 'O', 'M', 'P', 'A', 'N', 'G', 0, 's', 'h', 'a', 'l', 'l', 'o', 'w',
' ', 'd', 'r', 'u', 'm'
                    .db                 0x0A, 'K', 'O', 'M', 'P', 'L', 'O', 'T', 0, 'c', 'o', 'n', 's', 'p', 'i', 'r',
'a', 'c', 'y'
                    .db                 0x0A, 'K', 'O', 'N', 'G', 'K', 'O', 'N', 'G', 0, 'p', 'i', 'l', 'l', 'o', 'r',
'y'
                    .db                 0x0A, 'K', 'O', 'N', 'G', 'S', 'I', 0, 'a', 's', 's', 'o', 'c', 'i', 'a', 't',
'i', 'o', 'n', 'p', 'a', 'r', 't', 'n', 'e', 'r', 's', 'h', 'i', 'p'
                    .db                 0x0A, 'K', 'O', 'N', 'O', 'N', 0, 'i', 't', ' ', 'i', 's', ' ', 's', 'a', 'i',
'd'
                    .db                 0x0A, 'K', 'O', 'N', 'T', 'A', 'N', 'G', 0, 'd', 'r', 'y', 'p', 'a', 'r', 'c',
'h', 'e', 'd'
                    .db                 0x0A, 'L', 'A', 'J', 'U', 0, 'f', 'a', 's', 't'
                    .db                 0x0A, 'L', 'A', 'J', 'U', 'R', 0, 'c', 'o', 'l', 'u', 'm', 'n', ' ', '(', 'e',
'.', 'g', '.', ' ', 'o', 'f', ' ', 't', 'r', 'o', 'o', 'p', 's', ')'
                    .db                 0x0A, 'L', 'A', 'K', 'A', 'R', 0, 't', 'o', ' ', 's', 'k', 'e', 't', 'c', 'h',
't', 'o', ' ', 'o', 'u', 't', 'l', 'i', 'n', 'e'
                    .db                 0x0A, 'L', 'A', 'K', 'A', 'R', 'A', 'N', 0, 'a', ' ', 's', 'k', 'e', 't', 'c',
'h'
                    .db                 0x0A, 'L', 'A', 'K', 'I', 0, 'h', 'u', 's', 'b', 'a', 'n', 'd'
                    .db                 0x0A, 'L', 'A', 'K', 'I', '-', 'L', 'A', 'K', 'I', 0, 'm', 'a', 'l', 'e'
                    .db                 0x0A, 'L', 'A', 'K', 'N', 'A', 'T', 0, 't', 'o', ' ', 'd', 'a', 'm', 'n', 't',
'o', ' ', 'c', 'u', 'r', 's', 'e'
                    .db                 0x0A, 'L', 'A', 'K', 'O', 'N', 0, 't', 'o', ' ', 'a', 'c', 't', ' ', '(', 'i',
'n', ' ', 'a', ' ', 'p', 'l', 'a', 'y', ' ', 'o', 'r', ' ', 'f', 'i', 'l', 'm', ')'
                    .db                 0x0A, 'L', 'A', 'K', 'O', 'N', 'A', 'N', 0, 'a', 'n', ' ', 'a', 'c', 't', 'a',
' ', 'p', 'l', 'a', 'y', 'a', ' ', 'd', 'r', 'a', 'm', 'a'
                    .db                 0x0A, 'L', 'A', 'K', 'S', 'A', 0, 'a', ' ', 'p', 'r', 'e', 'p', 'a', 'r', 'a',
't', 'i', 'o', 'n', ' ', 'o', 'f', ' ', 'n', 'o', 'o', 'd', 'l', 'e', 's'
                    .db                 0x0A, 'L', 'A', 'K', 'S', 'A', 'M', 'A', 'N', 'A', 0, 'a', 'd', 'm', 'i', 'r',
'a', 'l'
                    .db                 0x0A, 'L', 'A', 'K', 'S', 'A', 'N', 'A', 0, 't', 'o', ' ', 'p', 'e', 'r', 'f',
'o', 'r', 'm', 't', 'o', ' ', 'e', 'x', 'e', 'c', 'u', 't', 'e', 't', 'o', ' ', 'a', 'c', 'c', 'o', 'm', 'p', 'l', 'i',
's', 'h'
                    .db                 0x0A, 'L', 'A', 'K', 'U', 0, '(', '1', ')', 't', 'o', ' ', 'h', 'a', 'p', 'p',
'e', 'n', 't', 'o', ' ', 'o', 'c', 'c', 'u', 'r', ' ', '(', '2', ')', 'm', 'a', 'r', 'k', 'e', 't', 'a', 'b', 'l', 'e'
                    .db                 0x0A, 'L', 'A', 'K', 'U', 'R', 0, 'f', 'u', 's', 'e', 'b', 'l', 'e', 'n', 'd'
                    .db                 0x0A, 'L', 'A', 'L', 'A', 0, 'a', ' ', 't', 'y', 'p', 'e', ' ', 'o', 'f', ' ',
's', 'e', 'a', ' ', 'm', 'o', 'l', 'l', 'u', 's', 'c'
                    .db                 0x0A, 'L', 'A', 'L', 'A', 'H', 0, 't', 'o', ' ', 'g', 'o', 'b', 'b', 'l', 'e'
                    .db                 0x0A, 'L', 'A', 'L', 'A', 'I', 0, 'c', 'a', 'r', 'e', 'l', 'e', 's', 's', 'n',
'e', 'g', 'l', 'i', 'g', 'e', 'n', 't', 'i', 'n', 'd', 'i', 'f', 'f', 'e', 'r', 'e', 'n', 't'
                    .db                 0x0A, 'L', 'I', 'M', 'B', 'U', 'N', 'G', 'A', 'N', 0, 'd', 'o', 'c', 'k', 'y',
'a', 'r', 'd'
```

```
                    .db                 0x0A, 'L', 'I', 'M', 'P', 'A', 0, 's', 'p', 'l', 'e', 'e', 'n'
                    .db                 0x0A, 'L', 'I', 'M', 'P', 'A', 'H', 0, 't', 'o', ' ', 'o', 'v', 'e', 'r', 'f',
'l', 'o', 'w'
                    .db                 0x0A, 'L', 'I', 'M', 'P', 'A', 'P', 0, '(', '1', ')', 'i', 'n', ' ', 'l', 'a',
'y', 'e', 'r', 's', ' ', '(', '2', ')', 'e', 'x', 'c', 'e', 's', 's', 'i', 'v', 'e', ' ', 'd', 'e', 'p', 't'
                    .db                 0x0A, 'L', 'I', 'N', 'A', 'N', 'G', 0, 't', 'o', ' ', 'f', 'l', 'o', 'w', ' ',
'(', 'f', 'o', 'r', ' ', 't', 'e', 'a', 'r', 's', ' ', 'o', 'r', ' ', 's', 'w', 'e', 'a', 't', ')'
                    .db                 0x0A, 'L', 'I', 'N', 'C', 'A', 'H', 0, 'n', 'i', 'm', 'b', 'l', 'e'
                    .db                 0x0A, 'L', 'I', 'N', 'C', 'I', 'R', 0, 's', 'm', 'o', 'o', 't', 'h', ' ', '(',
'i', 'n', ' ', 'm', 'o', 'v', 'i', 'n', 'g', ')'
                    .db                 0x0A, 'L', 'I', 'N', 'D', 'U', 'N', 'G', 0, 't', 'o', ' ', 't', 'a', 'k', 'e',
' ', 'r', 'e', 'f', 'u', 'g', 'e', ' ', 'o', 'r', ' ', 's', 'h', 'e', 'l', 't', 'e', 'r'
                    .db                 0x0A, 'L', 'I', 'N', 'D', 'U', 'N', 'G', 'A', 'N', 0, 'h', 'a', 'v', 'e', 'n',
'r', 'e', 'f', 'u', 'g', 'e'
                    .db                 0x0A, 'L', 'I', 'N', 'G', 'G', 'A', 0, 'c', 'o', 'l', 'o', 's', 's', 'u', 's'
                    .db                 0x0A, 'L', 'I', 'N', 'G', 'K', 'A', 'R', 0, 'a', ' ', 'c', 'o', 'i', 'l'
                    .db                 0x0A, 'L', 'I', 'N', 'G', 'K', 'A', 'R', 'A', 'N', 0, 'a', ' ', 'c', 'o', 'n',
'v', 'o', 'l', 'u', 't', 'i', 'o', 'n', 'a', ' ', 'c', 'o', 'i', 'l'
                    .db                 0x0A, 'L', 'I', 'N', 'G', 'K', 'U', 'N', 'G', 0, 't', 'o', ' ', 's', 'u', 'r',
'r', 'o', 'u', 'n', 'd', 't', 'o', ' ', 'e', 'n', 'c', 'i', 'r', 'c', 'l', 'e'
                    .db                 0x0A, 'L', 'I', 'N', 'G', 'K', 'U', 'N', 'G', 'A', 'N', 0, 'e', 'n', 'c', 'i',
'r', 'c', 'l', 'e', 'm', 'e', 'n', 't', 'e', 'n', 'c', 'l', 'o', 's', 'u', 'r', 'e', 's', 'p', 'h', 'e', 'r', 'e'
                    .db                 0x0A, 'L', 'I', 'N', 'G', 'K', 'U', 'P', 0, 'c', 'o', 'v', 'e', 'r'
                    .db                 0x0A, 'L', 'I', 'N', 'T', 'A', 'H', 0, 'l', 'e', 'e', 'c', 'h'
                    .db                 0x0A, 'L', 'I', 'N', 'T', 'A', 'N', 'G', 0, 'h', 'o', 'r', 'i', 'z', 'o', 'n',
't', 'a', 'l'
                    .db                 0x0A, 'L', 'I', 'N', 'T', 'A', 'N', 'G', '-', 'P', 'U', 'K', 'A', 'N', 'G', 0,
'h', 'e', 'l', 't', 'e', 'r', '-', 's', 'k', 'e', 'l', 't', 'e', 'r'
                    .db                 0x0A, 'L', 'I', 'N', 'T', 'A', 'S', 0, 't', 'o', ' ', 'p', 'a', 's', 's', ' ',
'b', 'y', 't', 'o', ' ', 'c', 'r', 'o', 's', 's'
                    .db                 0x0A, 'L', 'I', 'P', 'A', 'N', 0, 'c', 'e', 'n', 't', 'i', 'p', 'e', 'd', 'e'
                    .db                 0x0A, 'L', 'I', 'P', 'A', 'S', 0, 'c', 'o', 'c', 'k', 'r', 'o', 'a', 'c', 'h'
                    .db                 0x0A, 'L', 'I', 'P', 'A', 'T', 0, 't', 'o', ' ', 'f', 'o', 'l', 'd'
                    .db                 0x0A, 'L', 'I', 'P', 'A', 'T', 'A', 'N', 0, 'a', ' ', 'f', 'o', 'l', 'd'
                    .db                 0x0A, 'L', 'I', 'P', 'I', 'T', 0, 'n', 'a', 'r', 'r', 'o', 'w', ' ', 'h', 'e',
'm'
                    .db                 0x0A, 'L', 'I', 'P', 'U', 'R', 0, 't', 'o', ' ', 'c', 'o', 'm', 'f', 'o', 'r',
't', 't', 'o', ' ', 'c', 'o', 'n', 's', 'o', 'l', 'e'
                    .db                 0x0A, 'L', 'I', 'P', 'U', 'T', 0, 't', 'o', ' ', 'c', 'o', 'v', 'e', 'r', 't',
'o', ' ', 'c', 'o', 'm', 'p', 'r', 'i', 's', 'e'
                    .db                 0x0A, 'L', 'I', 'P', 'U', 'T', 'A', 'N', 0, 'c', 'o', 'v', 'e', 'r', 'a', 'g',
'e', 'i', 'n', 'c', 'l', 'u', 's', 'i', 'o', 'n'
                    .db                 0x0A, 'L', 'I', 'S', 'A', 'N', 0, 'o', 'r', 'a', 'l', 'v', 'e', 'r', 'b', 'a',
'l'
                    .db                 0x0A, 'L', 'I', 'S', 'U', 'T', 0, 's', 'h', 'r', 'u', 'n', 'k', 'e', 'n', 's',
'o', 'f', 't'
                    .db                 0x0A, 'L', 'I', 'T', 'A', 'R', 0, 'c', 'i', 'r', 'c', 'u', 'i', 't'
                    .db                 0x0A, 'L', 'I', 'T', 'U', 'P', 0, 'c', 'o', 'v', 'e', 'r', 'e', 'd', ' ', 'a',
'n', 'd', ' ', 'c', 'o', 'n', 'c', 'e', 'a', 'l', 'e', 'd'
                    .db                 0x0A, 'L', 'I', 'U', 'K', 0, 't', 'o', ' ', 's', 'n', 'a', 'k', 'e', 't', 'o',
' ', 's', 'w', 'a', 'y'
                    .db                 0x0A, 'L', 'I', 'W', 'A', 'T', 0, 's', 'o', 'd', 'o', 'm', 'y'
                    .db                 0x0A, 'L', 'O', 'B', 'A', 0, 'g', 'r', 'a', 's', 'p', 'i', 'n', 'g', 'g', 'r',
'e', 'e', 'd', 'y', ' ', 'f', 'o', 'r', ' ', 'm', 'o', 'n', 'e', 'y', ' ', 'o', 'r', ' ', 'p', 'o', 's', 's', 'e', 's',
's', 'i', 'o', 'n', 's'
                    .db                 0x0A, 'N', 'E', 'R', 'A', 'K', 'A', 0, 'h', 'e', 'l', 'l'
                    .db                 0x0A, 'N', 'G', 'A', 'N', 'G', 'A', 0, 'o', 'p', 'e', 'n', ' ', 't', 'h', 'e',
' ', 'm', 'o', 'u', 't', 'h', ' ', 'w', 'i', 'd', 'e'
                    .db                 0x0A, 'N', 'G', 'E', 'R', 'I', 0, 'a', 'p', 'p', 'a', 'l', 'l', 'i', 'n', 'g',
'e', 'e', 'r', 'i', 'e', 'h', 'o', 'r', 'r', 'i', 'f', 'y', 'i', 'n', 'g'
                    .db                 0x0A, 'N', 'G', 'I', 'A', 'U', 0, 'h', 'o', 'w', 'l', ' ', 'l', 'i', 'k', 'e',
' ', 'a', ' ', 'c', 'a', 't'
                    .db                 0x0A, 'N', 'G', 'U', 'A', 'K', 0, 'b', 'r', 'a', 'y', ' ', 'l', 'i', 'k', 'e',
' ', 'a', 'd', 'o', 'n', 'k', 'e', 'y'
                    .db                 0x0A, 'N', 'I', 'A', 'T', 0, 'i', 'n', 't', 'e', 'n', 't', 'i', 'o', 'n'
                    .db                 0x0A, 'N', 'I', 'K', 'M', 'A', 'T', 0, 'e', 'n', 'j', 'o', 'y', 'a', 'b', 'l',
'e', 'p', 'l', 'e', 'a', 's', 'i', 'n', 'g'
                    .db                 0x0A, 'N', 'I', 'L', 'A', 'I', 0, 'a', 's', 's', 'e', 's', 's', ' ', 't', 'h',
'e', ' ', 'v', 'a', 'l', 'u', 'e', ' ', 'o', 'f'
                    .db                 0x0A, 'N', 'I', 'L', 'A', 'M', 0, 's', 'a', 'p', 'p', 'h', 'i', 'r', 'e'
                    .db                 0x0A, 'N', 'I', 'R', 'I', 'N', 'S', 'A', 'N', 'I', 0, 'd', 'e', 'h', 'u', 'm',
'a', 'n', 'i', 'z', 'e'
                    .db                 0x0A, 'N', 'I', 'R', 'W', 'A', 'N', 'A', 0, 's', 't', 'a', 't', 'e', ' ', 'o',
'f', ' ', 'p', 'e', 'r', 'f', 'e', 'c', 't', ' ', 'b', 'l', 'i', 's', 's'
                    .db                 0x0A, 'N', 'I', 'S', 'A', 'N', 0, 'g', 'r', 'a', 'v', 'e', 's', 't', 'o', 'n',
'e'
                    .db                 0x0A, 'N', 'I', 'S', 'B', 'A', 'H', 0, 'r', 'a', 't', 'i', 'o'
                    .db                 0x0A, 'N', 'I', 'S', 'K', 'A', 'L', 'A', 0, 'h', 'a', 'v', 'i', 'n', 'g', ' ',
'n', 'o', ' ', 'm', 'a', 't', 'e', 'r', 'i', 'a', 'l', ' ', 'o', 'r', ' ', 'p', 'h', 'y', 's', 'i', 'c', 'a', 'l', ' ',
's', 'u', 'b', 's', 't', 'a', 'n', 'c', 'e'
                    .db                 0x0A, 'N', 'O', 'D', 'A', 0, 'd', 'i', 's', 'c', 'r', 'e', 'd', 'i', 't'
                    .db                 0x0A, 'N', 'O', 'K', 'T', 'A', 'H', 0, 'f', 'u', 'l', 'l', ' ', 's', 't', 'o',
'p'
                    .db                 0x0A, 'N', 'O', 'M', 'B', 'O', 'R', 0, 'n', 'u', 'm', 'b', 'e', 'r'
                    .db                 0x0A, 'P', 'A', 'K', 'A', 'I', 'A', 'N', 0, 'c', 'l', 'o', 't', 'h', 'e', 's'
                    .db                 0x0A, 'P', 'A', 'K', 'A', 'I', 'A', 'N', ' ', 'S', 'E', 'R', 'A', 'G', 'A',
'M', 0, 'u', 'n', 'i', 'f', 'o', 'r', 'm'
                    .db                 0x0A, 'P', 'A', 'K', 'A', 'L', 0, 't', 'o', ' ', 'c', 'a', 'u', 'l', 'k'
                    .db                 0x0A, 'P', 'A', 'K', 'A', 'R', 0, 'e', 'x', 'p', 'e', 'r', 't', 's', 'p', 'e',
'c', 'i', 'a', 'l', 'i', 's', 't'
                    .db                 0x0A, 'P', 'A', 'K', 'A', 'R', ' ', 'R', 'U', 'N', 'D', 'I', 'N', 'G', 0, 'c',
'o', 'n', 's', 'u', 'l', 't', 'a', 'n', 't'
                    .db                 0x0A, 'P', 'A', 'K', 'A', 'T', 0, 't', 'o', ' ', 'a', 'g', 'r', 'e', 'e', 't',
'o', ' ', 'c', 'o', 'n', 's', 'p', 'i', 'r', 'e'
                    .db                 0x0A, 'P', 'A', 'K', 'A', 'T', 'A', 'N', 0, 'p', 'a', 'c', 't', 'a', 'g', 'r',
'e', 'e', 'm', 'e', 'n', 't', 't', 'r', 'e', 'a', 't', 'y', 'c', 'o', 'n', 's', 'p', 'i', 'r', 'a', 'c', 'y'
                    .db                 0x0A, 'P', 'A', 'K', 'I', 'S', 0, 'f', 'e', 'r', 'n'
                    .db                 0x0A, 'P', 'A', 'K', 'S', 'A', 0, 't', 'o', ' ', 'f', 'o', 'r', 'c', 'e', 't',
'o', ' ', 'c', 'o', 'm', 'p', 'e', 'l'
                    .db                 0x0A, 'P', 'A', 'K', 'S', 'A', 'A', 'N', 0, 'c', 'o', 'm', 'p', 'u', 'l', 's',
'i', 'o', 'n'
                    .db                 0x0A, 'P', 'A', 'K', 'S', 'I', 0, 'a', 'x', 'i', 's'
                    .db                 0x0A, 'P', 'A', 'K', 'U', 0, 'n', 'a', 'i', 'l', 'f', 'e', 'r', 'n', 's'
                    .db                 0x0A, 'P', 'A', 'L', 'A', 0, 'n', 'u', 't', 'm', 'e', 'g'
                    .db                 0x0A, 'P', 'A', 'L', 'A', 'M', 0, '(', '1', ')', 'g', 'r', 'o', 'u', 't', ' ',
'(', '2', ')', 'p', 'l', 'u', 'g'
                    .db                 0x0A, 'P', 'A', 'L', 'A', 'N', 'G', 0, 'c', 'r', 'o', 's', 's', 'b', 'a', 'r'
                    .db                 0x0A, 'P', 'A', 'L', 'I', 'N', 'G', 0, '(', '1', ')', 't', 'h', 'e', ' ', 'm',
'o', 's', 't', 'e', 'x', 't', 'r', 'e', 'm', 'e', 'l', 'y', ' ', '(', '2', ')', 't', 'o', ' ', 't', 'u', 'r', 'n', ' ',
'a', 'w', 'a', 'y'
```

```
                .db                 0x0A, 'P', 'A', 'L', 'I', 'T', 0, 't', 'o', ' ', 's', 'm', 'e', 'a', 'r', 't',
'o', ' ', 's', 'm', 'u', 'd', 'g', 'e'
                .db                 0x0A, 'S', 'U', 'D', 'I', 0, 'w', 'i', 'l', 'l', 'i', 'n', 'g'
                .db                 0x0A, 'S', 'U', 'D', 'I', 'P', 0, 'w', 'o', 'o', 'd', 'e', 'n', ' ', 'l', 'a',
'd', 'l', 'e'
                .db                 0x0A, 'S', 'U', 'D', 'U', 0, 's', 'p', 'o', 'o', 'n'
                .db                 0x0A, 'S', 'U', 'D', 'U', 'T', 0, 'c', 'o', 'r', 'n', 'e', 'r', 'a', 'n', 'g',
'l', 'e', 'a', 's', 'p', 'e', 'c', 't', 'p', 'o', 'i', 'n', 't', ' ', 'o', 'f', ' ', 'v', 'i', 'e', 'w'
                .db                 0x0A, 'S', 'U', 'G', 'U', 'L', 0, 's', 'a', 'd', 'n', 'e', 's', 's', 'g', 'r',
'i', 'e', 'f', 's', 'o', 'r', 'r', 'o', 'w'
                .db                 0x0A, 'S', 'U', 'H', 'U', 0, 't', 'e', 'm', 'p', 'e', 'r', 'a', 't', 'u', 'r',
'e'
                .db                 0x0A, 'S', 'U', 'J', 'U', 'D', 0, 'p', 'r', 'o', 's', 't', 'r', 'a', 't', 'e',
' ', '(', 'i', 'n', ' ', 'p', 'r', 'a', 'y', 'e', 'r', ')'
                .db                 0x0A, 'S', 'U', 'K', 'A', 0, 'l', 'i', 'k', 'e', 'e', 'n', 'j', 'o', 'y'
                .db                 0x0A, 'S', 'U', 'K', 'A', ' ', 'H', 'A', 'T', 'I', 0, 'd', 'e', 'l', 'i', 'g',
'h', 't', 'e', 'd', 'h', 'a', 'p', 'p', 'y'
                .db                 0x0A, 'S', 'U', 'K', 'A', ' ', 'R', 'I', 'A', 0, 'g', 'a', 'y', 'h', 'a', 'p',
'p', 'y'
                .db                 0x0A, 'S', 'U', 'K', 'A', 'C', 'I', 'T', 'A', 0, 'g', 'l', 'a', 'd', 'h', 'a',
'p', 'p', 'y', 'c', 'h', 'e', 'e', 'r', 'f', 'u', 'l'
                .db                 0x0A, 'S', 'U', 'K', 'A', 'N', 0, 's', 'p', 'o', 'r', 't'
                .db                 0x0A, 'S', 'U', 'K', 'A', 'R', 0, 'd', 'i', 'f', 'f', 'i', 'c', 'u', 'l', 't',
'h', 'a', 'r', 'd'
                .db                 0x0A, 'S', 'U', 'K', 'A', 'R', 'E', 'L', 'A', 0, 'v', 'o', 'l', 'u', 'n', 't',
'a', 'r', 'y'
                .db                 0x0A, 'S', 'U', 'K', 'A', 'R', 'E', 'L', 'A', 'W', 'A', 'N', 0, 'v', 'o', 'l',
'u', 'n', 't', 'e', 'e', 'r'
                .db                 0x0A, 'S', 'U', 'K', 'A', 'T', 0, 'q', 'u', 'a', 'n', 't', 'i', 't', 'y', 'c',
'a', 'p', 'a', 'c', 'i', 't', 'y'
                .db                 0x0A, 'S', 'U', 'K', 'A', 'T', 'A', 'N', 0, 'm', 'e', 'a', 's', 'u', 'r', 'e',
'm', 'e', 'n', 't', 'a', 's', 's', 'e', 's', 's', 'm', 'e', 'n', 't'
```

Main Malay Language Toolkit Code

The following is the code for the main project.

```
.include "4414mac.asm"


;REGISTER DEFINITIONS
.def            CurrAddr0   = r1
.def            CurrAddr1   = r2
.def            rTemp               = r16
.def            rTemp2              = r17
.def            rLookupFlag = r18


;Exclusively for Keyboard/LCD/EEPROM functions
.def            rLCDcharcnt = r19               ;LCD
.def            rE256k_ADD0 = r20               ;EEPROM
.def            rE256k_ADD1 = r21               ;EEPROM
.def            rLine               = r22       ;KEYBOARD

;Exclusively for Dictionary, rMalay used in Hangman
.def            rMalay              = r23       ;Char in Malay
.def            rDict               = r24       ;Char in Dict


;Exclusively for Hangman
.def            rScore              = r25       ;Score 0-6
.def            PWinFlag    = r26               ;Potentially winning
.def            MaskZL              = r27
.def            MaskZH              = r28
.def            Feature     = r29


.equ            QUERY           ='?'
.equ            NEWLINE         ='*'
.equ            GOBACK  ='#'
.equ            DELIM1          =':'
.equ            DELIM3          =0x0A
.equ            HANGMAN ='H'
.equ            DICT    ='D'
.equ            LEARN           ='L'
.equ            BEGINHANG='o'
.equ            BEGINDICT='!'
.equ            BEGINLEARN ='l'

.dseg
                Malay:                              .BYTE 16
                English:        .BYTE 16
                Target:                             .BYTE 16
                Mask:                               .BYTE 16
                LearnEnglish:   .BYTE 16

.cseg

.org $0000
                rjmp            RESET
```

```
                reti
                reti
                reti
                reti
                reti
                reti
                rjmp            lcd16_timer0
                reti
                reti
                reti
                reti
                reti

Greeting:       .db             "Welcome (D/H/L?)",0
WelDict:        .db             "Dictionary->Star", 0
Lookingup:      .db             "Checking...",0
LookupFail:     .db             "Can't find it =(",0
WelHangman:     .db             "Hangman->Head",0
WelLearn:       .db             "Learn ->Bulb",0
Congrat:        .db             "Won =) More(Y)?",0
Sorry:                          .db             "Lost =( More(Y)?",0
AnswerIs:       .db             "Ans:",0


RESET:
                setStack
                sei
                rcall           lcd16_init
                ser             r16
                out             DDRB,r16
                out             PORTB,r16

StartTimer:
                ldi             r16,0b00000001 ;Timer1 ticking
                out             TCCR1B,r16

InitAll:
                rcall           lcd16_clr
                flashZ          Greeting
                rcall           lcd16_printFlash
                rcall           k48_getKey
                rcall           k48_btnNum
                rcall           k48_Num2Char
                cpi             r16,DICT
                breq            GoDict
                cpi             r16,HANGMAN
                breq            GoHangman
                cpi             r16,LEARN
                breq            GoLearn
                rjmp            InitAll
GoDict:
                ldi             Feature,1
                rjmp            InitDict
GoLearn:
                rjmp            InitLearn
GoHangman:
                rjmp            InitHangman

;================================Learn Section=========================
InitLearn:
                ldi             Feature,2
                rcall           lcd16_clr
                flashZ          WelLearn
                rcall           lcd16_printFlash
                rcall           k48_getKey
                rcall           k48_btnNum
                rcall           k48_Num2Char
                cpi             r16,BEGINDICT
                breq            STARTLEARN
                rjmp            InitHangman

STARTLEARN:
                rcall           GetEnglishTargetMask
                ;Properly Initialize LearnEnglish,
                ;Target and Mask
;               ;************TEST***************/
;               rcall           lcd16_clr
;               ramZ            Target
;               rcall           lcd16_printRAM
;               ldi             rTemp,'&'
;               rcall           lcd16_putc

;               rcall           k48_getKey
;               rcall           lcd16_clr
;               ramZ            Mask
;               rcall           lcd16_printRAM
```

```
;               ldi             rTemp,'&'
;               rcall           lcd16_putc
                rcall           k48_getKey


DISPLAYENG:
                ramZ            LearnEnglish
;               rcall           lcd16_clr
;               rcall           lcd16_printRAM
;               rcall           k48_getKey
                rcall           lcd16_clr
                clr             rTemp2

LearnForEachLCDEntry:
                cpi             rTemp2,15       ;check if displayed characters>=15
                brge            BlockForEachLine;Block until newline key is pressed
                ld              r16,Z+                          ;Contains the ith
LearnEnglish char
;               mov             r16,r0
                cpi             r16, 0xFF       ;Check if end of file
                breq            LearnEnd
                cpi             r16,0x0A        ;Check if this is the last character
                breq            LearnEnd                        ;start over
                cpi             r16,0x00
                breq            LearnEnd
                rcall           lcd16_putc      ;display to LCD
                inc             rTemp2                          ;increment character count
                rjmp            LearnForEachLCDEntry

BlockForEachLine:
                ldi             rTemp, 0x3E
                rcall           lcd16_putc
                rcall           k48_getKey
                rcall           k48_btnNum
                rcall           k48_Num2Char
                cpi             r16,'*'
                brne            BlockForEachLine
                clr             rTemp2                          ;if usr pressed "NEWLINE"
                rcall           lcd16_clr
                rjmp            LearnForEachLCDEntry
LearnEnd:
                rcall           k48_getKey
                rcall           k48_btnNum
                rcall           k48_Num2Char
                cpi             r16,GOBACK
                breq            LearnScrollBack
                rjmp            HangmanLearnMode

LearnScrollBack:
                rcall           lcd16_clr
                clr             rTemp2
                ramZ            LearnEnglish
                rjmp            LearnForEachLCDEntry

HangmanLearnMode:
                rcall           lcd16_clr
                ramZ            Mask
                rcall           lcd16_printRAM                  ;Print "_ _ _"
                ldi             rScore, 10

;rScore contains the score
;So EnterGuessC runs at most 6 times


LearnEnterGuessC:
                cpi             rScore, 0
                breq            EndLearningGame
                rcall           k48_getKey
                rcall           k48_btnNum
                rcall           k48_Num2Char
                ;r16 contais the guessed character
                rcall           MatchC
                cpi             r17,0xff        ;User got a good characeter
                breq            CheckGood       ;Check if he has won already
                dec             rScore
                rjmp            LearnEnterGuessC

EndLearningGame:
                strcmpRAM       Target, Mask
                cpi             r16, 1
                breq            Good
                rjmp            Bad

                ;End of 20 trials, either good/bad student

CheckGood:
```

```
                ;If the entire target is 0xff, then he wins
                ;Otherwise give him more chances

                strcmpRAM       Target, Mask
                cpi             r16, 1
                breq            Good
                rjmp            LearnEnterGuessC

Good:
                flashZ          Congrat
                rcall           lcd16_clr
                rcall           lcd16_printFlash
                rjmp            ENDLEARNING

Bad:
                flashZ          Sorry
                rcall           lcd16_clr
                rcall           lcd16_printFlash
                rcall           k48_getKey
                rcall           lcd16_clr
                flashZ          AnswerIs
                rcall           lcd16_printFlash
                ramZ            Target
                rcall           lcd16_printRAM
                rcall           k48_getKey
                rjmp            ENDLEARNING


ENDLEARNING:
                rcall           k48_getKey
                rcall           k48_btnNum
                rcall           k48_Num2Char
                cpi             r16,'Y'
                breq            RESET_LEARN     ;If more Hangman, go to Hangman/Learning
Section
                rjmp            InitAll                         ;Else go back to first
Welcome screen
RESET_LEARN:
                rjmp            STARTLEARN


;=================================Hangman Section=====================
InitHangman:
                ldi             Feature, 3
                rcall           lcd16_clr
                flashZ          WelHangman
                rcall           lcd16_printFlash
                rcall           k48_getKey
                rcall           k48_btnNum
                rcall           k48_Num2Char
                cpi             r16,BEGINHANG
                breq            STARTHANGMAN
                rjmp            InitHangman

STARTHANGMAN:
                rcall           GetATargetMask
                rcall           lcd16_clr
                ramZ            Mask
                rcall           lcd16_printRAM                  ;Print "_ _ _"
                ldi             rScore, 6
                rjmp            PrintScore
PrintScore:
                ldi             r16,'('                                         ;Print Score
                rcall           lcd16_putc
                mov             r16,rScore
                addi            r16,'0'
                rcall           lcd16_putc
                ldi             r16,')'
                rcall           lcd16_putc

;rScore contains the score
;So EnterGuessC runs at most 6 times


EnterGuessC:
                ldi             r16, 0b11111111
                out             PORTB, r16      ;************
                cpi             rScore, 0
                breq            EndGame
                rcall           k48_getKey
                rcall           k48_btnNum
                rcall           k48_Num2Char
                ;r16 contais the guessed character
                rcall           MatchC
                cpi             r17,0xff        ;User got a good characeter
                breq            CheckWin        ;Check if he has won already
```

```
                dec             rScore
                ldi             r16,'('                                         ;Print Score
                rcall           lcd16_putc
                mov             r16,rScore
                addi            r16,'0'
                rcall           lcd16_putc
                ldi             r16,')'
                rcall           lcd16_putc
                rjmp            EnterGuessC

EndGame:
                strcmpRAM       Target, Mask
                cpi             r16, 1
                breq            Won
                rjmp            Lost

                ;End of 6 trials, either win/lose


CheckWin:

                ;If the entire target is 0xff, then he wins
                ;Otherwise give him more chances
                ldi             r16,'('                                         ;Print Score
                rcall           lcd16_putc
                mov             r16,rScore
                addi            r16,'0'
                rcall           lcd16_putc
                ldi             r16,')'
                rcall           lcd16_putc
                strcmpRAM       Target, Mask
                cpi             r16, 1
                breq            Won
                rjmp            EnterGuessC

Won:
                flashZ          Congrat
                rcall           lcd16_clr
                rcall           lcd16_printFlash
                rjmp            ENDHANGMAN

Lost:
                flashZ          Sorry
                rcall           lcd16_clr
                rcall           lcd16_printFlash
                rcall           k48_getKey
                rcall           lcd16_clr
                flashZ          AnswerIs
                rcall           lcd16_printFlash
                ramZ            Target
                rcall           lcd16_printRAM
                rcall           k48_getKey
                rjmp            ENDHANGMAN


ENDHANGMAN:
                rcall           k48_getKey
                rcall           k48_btnNum
                rcall           k48_Num2Char
                cpi             r16,'Y'
                breq            RESET_0                         ;If more Hangman, go to
Hangman/Learning Section
                rjmp            InitAll                         ;Else go back to first
Welcome screen
RESET_0:
                rjmp            STARTHANGMAN

;=====================END of Hangman Section=======================



;===========Step1: Display greeting==================================
InitDict:
                rcall           lcd16_clr
                flashZ          WelDict
                rcall           lcd16_printflash
                ldi     rE256k_ADD0,LOW(2*FakeEEPROM)
        ldi      rE256k_ADD1,HIGH(2*FakeEEPROM)

===========Step2: Record Lookup Malay Word=========================
StartRecord:
                rcall           k48_getKey
                rcall           k48_btnNum
                rcall           k48_Num2Char
                cpi             r16,BEGINDICT  ;Detect BEGIN Key
                brne            StartRecord
```

```
                ldi             rTemp, 1
                com             rTemp
                out             PORTB,rTemp     ;LED = 0000 0001 (Got BEGIN)
                rcall           lcd16_clr       ;Clear up LCD for UserInput
                ramZ            Malay                           ;set up pointer to Malay RAM


Record:         rcall           k48_getKey      ;Poll for KeyPress
                rcall           k48_btnNum      ;Translate Keycode to ButtonNumber
                rcall           k48_Num2Char    ;Translate ButtonNumber to ASCII char

                push            r16
                cpi             r16,QUERY       ;Check if Malay word has ended
                breq            EndRecord
                rcall           lcd16_putc      ;Display the entered character

                pop             r16
                st              Z,r16                           ;Store the entered character
into
                                                                ;allocated RAM for Malay

                adiw            ZL,1                            ;Increment the Memory address
                ;clr            rTemp
                ;adc            ZH,rTemp
                rjmp            Record                          ;Loop back

EndRecord:
                ;Null-terminate the Malay string
                rcall           lcd16_putc      ;Display the entered character
                clr             rTemp
                st              Z, rTemp        ;Add the Null character
                ldi             rTemp, 2
                com             rTemp
                out             PORTB,rTemp     ;LED =          0000 00010 (Got QUERY)



==========Step3: Lookup English Word==============================
SearchStart:
                ldi             ZL,LOW(Malay)
                ldi             ZH,HIGH(Malay)

ForEachNewLine:

                rcall           e256k_getc
                cpi             r16, 0xFF       ;Check if end of file
                breq            EndAllSearch
                cpi             r16, 0x0A
                breq            ForEachChar     ;Found a newline character =)
                rjmp            ForEachNewLine

ForEachChar:
                ldi             rTemp,0b00000100                ;LED says I found an entry
                com             rTemp
                out             PORTB, rTemp
                rcall           e256k_getc      ;r16 now contains the ith Malay char
                                                                ;in the Dictionary in current
line
                mov             rDict,r16                       ;r20 = char in dictionary
                cpi             rDict, 0xFF     ;Check if end of file
                breq            EndAllSearch
                ld              r0,Z+                           ;r0 now contains the ith
Malay char
                                                                ;to be looked up
                mov             rMalay,r0                       ;r21 = char in the Malay
                cp              rDict,rMalay                    ;compare the two characters
                breq            PSearchHit      ;the current 2 chacters match
                clr             rLookupFlag     ;didn't match when neither word has reached
the end
                rjmp            SearchStart     ;skip to the next Malay:English entry in
Dictionary


PSearchHit:
                ldi             rTemp, 0b00001000               ;Potential Search Hit
                com             rTemp
                out             PORTB, rTemp

                ser             rLookupFlag
                cpi             rDict, 0                        ;check if this is the end of
Dict
                                                                ;Malay word
                breq            SearchHit       ;DictMalay = '\0', LookupMalay ='\0'
                rjmp            ForEachChar     ;Either word not Ended yet, continue
```

```
matching

SearchMiss:                                     ;Search Miss for current line
                clr             rLookupFlag
                rjmp            ForEachLine

EndAllSearch:
                rcall           lcd16_clr
                flashZ          LookupFail
                rcall           lcd16_printflash
                rjmp            END

=================Step 4:Display to LCD English Word=====================
SearchHit:
                ;Load the English Word to RAM and eventually to LCD
                ldi             rTemp,4
                com             rTemp
                out             PORTB,rTemp     ;LED = 0000 1000 for a search hit
                clr             rTemp2                          ;temp var that holds the ith
LCD display line

                mov             CurrAddr0,rE256k_Add0           ;Remember where the English
Translation begins
                mov             CurrAddr1,rE256k_Add1
                rcall           lcd16_clr

ForEachLCDEntry:
                cpi             rTemp2,15       ;check if displayed characters>=15
                brge            ForEachLine     ;Block until newline key is pressed
                rcall           e256k_getc      ;r16 holds an English Character
                cpi             r16, 0xFF       ;Check if end of file
                breq            End
                cpi             r16,0x0A        ;Check if this is the last character
                breq            End                             ;start over
                cpi             r16,0x00
                breq            End
                rcall           lcd16_putc      ;display to LCD
                inc             rTemp2                          ;increment character count
                rjmp            ForEachLCDEntry

ForEachLine:
                ldi             rTemp, 0x3E
                rcall           lcd16_putc
                rcall           k48_getKey
                rcall           k48_btnNum
                rcall           k48_Num2Char
                cpi             r16,'*'
                brne            ForEachLine
                clr             rTemp2                          ;if usr pressed "NEWLINE"
                rcall           lcd16_clr
                rjmp            ForEachLCDEntry
End:
                rcall           k48_getKey
                rcall           k48_btnNum
                rcall           k48_Num2Char
                cpi             r16,GOBACK
                breq            ScrollBack
                rjmp            InitAll

ScrollBack:
                rcall           lcd16_clr
                clr             rTemp2
                mov             rE256k_Add0,CurrAddr0
                mov             rE256k_Add1,CurrAddr1
                rjmp            ForEachLCDEntry
========================================================================




;---------------------------------------
;INCLUDE FILES AND REQUIRED CONSTANTS

;---------------------------------------
;12x4 Keyboard
;INPUT:
;  constants to be defined:
;               k48_PORT1; the port used for controlling the keyboard (.e.g .equ
k48_PORT1=PORTA)
;               k48_PORT2; the port used for controlling the keyboard (.e.g .equ
k48_PORT2=PORTD)
;               k48_PIN1 ; the pin used for controlling the keyboard (e.g. .equ
k16_PIN1=PINA)
;               k48_PIN2 ; the pin used for controlling the keyboard (e.g. .equ
k16_PIN2=PIND)
```

```
;              k48_DD1  ; for setting the Data-Direction (e.g. .equ k48_DD1=DDRA)
;              k48_DD2  ; for setting the Data-Direction (e.g. .equ k48_DD2=DDRD)
;  registers we use:
;              rLine
;  internal function:
;              checkLine
;For Keyboard
.equ           k48_PORT1 = PORTA
.equ           k48_PORT2 = PORTC
.equ           k48_PIN1  = PINA
.equ           k48_PIN2  = PINC
.equ           k48_DD1          = DDRA
.equ           k48_DD2   = DDRC
.include "kb48mac.asm"

;-------------------------------------
;16x1 LCD
;  constants to be defined:
;              lcd16_DD       ; for setting the Data-Direction (e.g. .equ lcd16_DD=DDRD)
;              lcd16_PORT     ; the port used for controlling the keyboard (.e.g .equ
lcd16_PORT=PORTD)
;              lcd16_PIN      ; the pin used for controlling the keyboard (e.g. .equ
lcd16_PIN=PIND)
;              lcd16_rs       ; the LCD rs pin number (e.g. .equ lcd16_rs=PD6)
;              lcd16_rw       ; the LCD rw pin number (e.g. .equ lcd16_rw=PD5)
;              lcd16_en       ; the LCD en pin number (e.g. .equ lcd16_en=PD4)
;  registers to be defined:
;              rLCDcharcnt ;to hold the current char location
;  1. timer0 overflow interrupt must call lcd16_timer0 when LCD is used (say, implement
states)
;  2. after finish resetting, and sei, must call lcd16_init
.equ           lcd16_DD = DDRD
.equ           lcd16_PORT = PORTD
.equ           lcd16_PIN  = PIND
.equ           lcd16_rs   = PD6
.equ           lcd16_rw   = PD5
.equ           lcd16_en   = PD4
.include "lcd16mac.asm"

;********************GetATargetMask**Begin*******************/

GetATargetMask:
               push           r16
               push           r17
               in             r16,TCNT1L     ;r16 contains a random number
               in             r17,TCNT1H     ;r17 contains another rand
;              com            r17
;              andi           r17,0b00000001 ;now r16 and r17 give me the

               andi           r16,0b11111111

               ldi            r17,0
;              ldi            r16,0

               ldi            rE256k_ADD0,LOW(2*FakeEEPROM)
               ldi            rE256k_ADD1,HIGH(2*FakeEEPROM)
               add            rE256k_ADD0,r16
               add            rE256k_ADD1,r17

loopHead:
               rcall          e256k_getc     ;r16 contains the random char
               cpi            r16,0x0A       ;check if it's beginning of Malay
                                                    ;word
               breq           CopyMalay
               rjmp           loopHead

;Copy Malay word from flash to Target and Mask in RAM
;load Z for Target
;load Z for Mask
;while(e256_getc !=0){
;              *Target = r16;
;              *Mask = '_';
;              ZTarget++;
;              ZMask++;
;}

CopyMalay:
               ramZ           Target         ;ZL and ZH holds address for Target
               ldi            MaskZL,LOW(Mask)
               ldi            MaskZH,HIGH(Mask);similar for Mask
ForEachC:
               rcall          e256k_getc
               st             Z+,r16                        ;Store char in Target

               push           ZL
               push           ZH
```

```
                mov             ZL,MaskZL
                mov             ZH,MaskZH
                cpi             r16, 0
                ifeq            1
                                ldi             r17, 0
                cpi             r16, 0
                ifne            1
                                ldi             r17,'_'
                st              Z+,r17                          ;Store '_' in Mask
                mov             MaskZL, ZL
                mov             MaskZH, ZH
                pop             ZH
                pop             ZL

                cpi             r16,0
                breq            EndGetTargetMask
                rjmp            ForEachC
EndGetTargetMask:
                pop             r17
                pop             r16
                ret
;********************GetATargetMask**End*******************/


;********************MatchC*********************************/
;Input: r16 stores the Character to be matched,Target stores the Word guessed
;Output:r17 stores the Match(0xff)/NoMatch(0x00)Information
;               Printing to LCD screen '_' or matched character until the end of word
;               Printing to LCD screen current score
;               Target's matched characters will be changed to char in Mask's
corresponding
;               position
;rMalay is destroyed

MatchC:
                ldi             r17,0x00                        ;No Match to begin with
                ramZ            Target                                  ;Load Target
                ldi             MaskZL, LOW(Mask)               ;Load Mask
                ldi             MaskZH, HIGH(Mask)
                rcall           lcd16_clr

TraverseTarget:
                ld              r0,Z                                    ;now R0
contains the word in Target
                mov             rMalay,r0
                cpi             rMalay,0                        ;check if end of Target

                breq            EndMatchC
                cp              rMalay,r16                      ;match
                breq            CMatch
                push            r16
;               ldi             r16,'_'
;               rcall           lcd16_putC                      ;Print the '_'
                adiw            ZL,1                                    ;go to the
next character
                ldi             r16,0
                inc             MaskZL
                adc             MaskZH,r16                      ;go to next mask position
                pop             r16
                rjmp            TraverseTarget

EndMatchC:
                ramZ            Mask
                rcall           lcd16_printRAM                  ;Print MASK
                ret
CMatch:
                ser             r17                                     ;Set the Match
bit
                push            r16

                ldi             r16, 0b01010101
                out             PORTB, r16      ;************

                mov             r16,rMalay

                push            ZL                                      ;
                push            ZH                                      ;
                mov             ZL,MaskZL                       ;
                mov             ZH,MaskZH                       ;
                st              Z,r16                                   ;Replace the _
to Character in Mask
                pop             ZH                                      ;

                pop             ZL                                      ;

                adiw            ZL,1                                    ;go to the
```

```
next character
                ldi             r16,0
                inc             MaskZL
                adc             MaskZH,r16                      ;go to next mask position

                ldi             r16, 0b00001111
                out             PORTB, r16      ;************

                pop             r16
                rjmp            TraverseTarget
;*************************************************************
;********************GetEnglishTargetMask**Begin*******************/

GetEnglishTargetMask:
                push            r16
                push            r17
                in              r16,TCNT1L      ;r16 contains a random number
                in              r17,TCNT1H      ;r17 contains another rand
;               com             r17
;               andi            r17,0b00000001 ;now r16 and r17 give me the

                andi            r16,0b11111111

                ldi             r17,0
;               ldi             r16,0

                ldi             rE256k_ADD0,LOW(2*FakeEEPROM)
                ldi             rE256k_ADD1,HIGH(2*FakeEEPROM)
                add             rE256k_ADD0,r16
                add             rE256k_ADD1,r17

loopMalay:
                rcall           e256k_getc      ;r16 contains the random char
                cpi             r16,0x0A        ;check if it's beginning of Malay
                                                ;word
                breq            CopyEnglish
                rjmp            loopMalay

;Copy Malay word from flash to Target and Mask in RAM
;load Z for Target
;load Z for Mask
;while(e256_getc !=0){
;               *Target = r16;
;               *Mask = '_';
;               ZTarget++;
;               ZMask++;
;}

CopyEnglish:
                ramZ            Target          ;ZL and ZH holds address for Target
                ldi             MaskZL,LOW(Mask)
                ldi             MaskZH,HIGH(Mask);similar for Mask
ForEachMC:
                rcall           e256k_getc
                st              Z+,r16                          ;Store char in Target

                push            ZL
                push            ZH
                mov             ZL,MaskZL
                mov             ZH,MaskZH
                cpi             r16, 0
                ifeq            1
                                ldi             r17, 0
                cpi             r16, 0
                ifne            1
                                ldi             r17,'_'
                st              Z+,r17                          ;Store '_' in Mask
                mov             MaskZL, ZL
                mov             MaskZH, ZH
                pop             ZH
                pop             ZL

                cpi             r16,0
                breq            EndGetETargetMask
                rjmp            ForEachMC

EndGetETargetMask:
                ;r256k_ADD0 and ADD1 are pointing to the English Explanation
                ramZ            LearnEnglish

ForEachEC:
                rcall           e256k_getc
                cpi             r16,0x0A
                breq            EndGetEnglishTargetMask
                st              Z+,r16                          ;Store char in Target
                rjmp            ForEachEC
```

```
EndGetEnglishTargetMask:
                ldi             r16,0                           ;Null terminate LearnEnglish
                st              Z+,r16
                pop             r17
                pop             r16
                ret
;************************************************************
;
;
;
;
;
;_______________________________________
;256k EEPROM Functions (Atmel AT24C256)
;
;INPUT:
;  constants to be defined:
;               e256k_DD        ; for setting the Data-Direction (e.g. .equ e256k_DD=DDRA)
;               e256k_PORT      ; the port used for controlling the EEPROM (e.g. .equ
e256k_PORT=PORTA)
;               e256k_PIN       ; the pin/port used for controlling the EEPROM (.e.g .equ
e256k_PIN=PINA)
;               e256k_SCL       ; the pin used for controlling the EEPROM's SCL pin (e.g.
.equ e256k_SCL=1)
;               e256k_SDA       ; the pin used for controlling the EEPROM's SDA pin (e.g.
.equ e256k_SDA=2)
;
;REGISTERS:
;  rE256k_ADD0, rE256k_ADD1: current word address
;
;EEPROM Connection
;EEpin  Connection
;__________________
;1 - A0 -- Currently not used, GND
;2 - A1 -- Currently not used, GND
;3 - NC
;4 - GND
;5 - SDA
;6 - SCL
;7 - WP (Write Protect, 5V to write protect)
;8 - VCC
.equ            e256k_DD  = DDRB
.equ            e256k_PORT= PORTB
.equ            e256k_PIN = PINB
.equ            e256k_SCL = PD1
.equ            e256k_SDA = PD2
.include "eepstub.asm"
```