

Interface Control Document

19/04/00 9:56:37

19/04/00 14:16:58

30/04/00 12:36:36

9/05/00 14:47:36

4/06/00 0:20:07

Keypad input:

Keypad input shall be handled by an interrupt service routine KB_IN_ISR.

It shall use the EXTINT signal on **IO_PORT2.2** to generate the interrupt from the data ready signal.

It shall use **IO_PORT1.0** to **IO_PORT1.3** for data bits. Other bits will be read but not used. **All function calls must be interruptible or make provisions to disable interrupts.**

During the service routine all interrupts will be disabled. Data shall be stored in the data buffer KB_STR

The ISR shall maintain **three** registers:

KB_CNT

Size: word

Purpose: holds the index to the last character in the string. The first character's index is 0.

Initialisation: set to 0

KBBYTE

Size: byte

Purpose: working register for data processing

Initialisation: not initialised

KBCHAR

Size: byte

Purpose: working register for data processing

Initialisation: not initialised

The ISR shall use one memory buffer:

KB_STR

Size: 16 bytes

Purpose: holds the character string most recently read from the keypad. The first location holds the current length of the string.

Data: Characters shall be ASCII coded.

Initialisation: in the startup file

Termination: string shall be NULL terminated

The ISR shall reset KB_CNT.

The ISR shall not preserve the contents of KBBYTE

The ISR shall not preserve the contents of KBCHAR

Serial port input:

Keypad input shall be handled by an interrupt service routine PC_IN_ISR.

It shall use the RI to generate the interrupt.

It shall read data from SBUF. During the service routine all interrupts will be disabled. Data will be stored in the data buffer PC_STR.

The ISR shall maintain **three** registers:

PC_CNT

Size: word

Purpose: holds the index to the last character in the string. The first character's index is 0.

Initialisation: set to 0

PCBYTE

Size: byte

Purpose: working register for data processing

Initialisation: not initialised

PCCHAR

Size: byte

Purpose: working register for data processing

Initialisation: not initialised

The ISR shall use one memory buffer:

PC_STR

Size: 16 bytes

Purpose: holds the character string most recently read by from the serial port. The first location holds the current length of the string.

Data: Characters shall be ASCII coded.

Initialisation: in the startup file

Termination: string shall be NULL terminated

The ISR shall reset PC_CNT.

The ISR shall not preserve the contents of PCBYTE

The ISR shall not preserve the contents of PCCHAR

PC serial port output:

Serial port output shall be handled by the subroutine TM_OUT.

It shall use SBUF to send data bytes out.

During the service routine all interrupts will be disabled.

The subroutine shall maintain three registers:

TM_PEND

size: word

purpose: holds the address of the character string to be transmitted

Initialization: not initialized.

TM_CNT

Size: word

Purpose: holds the index to the current character in the string. The first character's index is 0.

Initialization: set to 0

TMBYTE

Size: byte

Purpose: working register for data processing

Initialization: not initialized.

TMCHAR

Size: byte

Purpose: working register for data processing

Initialization: not initialized.

The subroutine shall use this memory buffer format:

Size: not limited

Purpose: holds the character string to be transmitted by the serial port. The first location holds the first character of the string.

Termination: the string shall be NULL terminated

The subroutine shall modify TM_PEND.

The subroutine shall not preserve the contents of TMBYTE

The subroutine shall not preserve the contents of TMCHAR

LCD output:

LCD output shall be handled by the subroutines LCD_OUT1 and LCD_OUT2.

they shall use IOPORT1 to send data bytes out in 4 bit mode.

During the function calls all interrupts will be disabled.

The subroutine shall maintain three registers:

LCD_PEND

size: word

purpose: holds the address of the character string to be transmitted

Initialization: not initialized.

CHAR_OUT

Size: word

Purpose: holds the index to the current character in the string. The first character's index is 0.

Initialization: set to 0

LCDBYTE

Size: byte

Purpose: working register for data processing

Initialization: not initialized.

The subroutine shall use this memory buffer format:

Size: limited to 40 characters else data will overwrite.

Purpose: holds the character string to be transmitted by the serial port. The first location holds the first character of the string.

Termination: the string shall be NULL terminated

The subroutine shall modify LCD_PEND.

The subroutine shall not preserve the contents of LCDBYTE

The subroutine shall modify CHAR_OUT

Provisional definitions:

```
;IO port P1 masks and signals
;
ASCII_NUM      EQU    48          ; ASCII conversion number
HASH           EQU    16?        ; hash key value
NULL           EQU    0          ; ASCII NULL value

ECHO_ON        EQU    00000010b   ; mask bit to OR ECHO high
ECHO_OFF       EQU    11111101b   ; mask bit to AND ECHO low
ECHO           EQU    1          ; bit number for ECHO signal(SC_STAT)

KB_BUSYH       EQU    00000100b   ; mask bit to OR KB_BUSY high
KB_BUSYL       EQU    11111011b   ; mask bit to AND KB_BUSY low
KB_BUSY        EQU    2          ; bit number for KB_BUSY signal(SC_STAT)

PC_BUSYH       EQU    00001000b   ; mask bit to OR PC_BUSY high
PC_BUSYL       EQU    11110111b   ; mask bit to AND PC_BUSY low
PC_BUSY        EQU    3          ; bit number for PC_BUSY signal(SC_STAT)
```

```

CRET                EQU    13            ; check this value

LCD_BUSYH           EQU    00010000b    ; mask bit to OR LCD_BUSY high
LCD_BUSYL           EQU    11101111b    ; mask bit to AND LCD_BUSY low
LCD_BUSY            EQU    4            ; bit number for LCD_BUSY signal(SC_STAT)

TM_BUSYH            EQU    00100000b    ; mask bit to OR TM_BUSY high
TM_BUSYL            EQU    11011111b    ; mask bit to AND TM_BUSY low
TM_BUSY             EQU    5            ; bit number for TM_BUSY signal(SC_STAT)

KB_MODEH            EQU    10000000b    ; mask bit to OR KB_MODE high
KB_MODEL            EQU    01111111b    ; mask bit to AND KB_MODE low
KB_MODE             EQU    7            ; bit number for KB_MODE signal(SC_STAT)

```

Menu selections:

- 1: weigh
- 2: tare
- 3: count
- 4: calibrate
- *: reset

Editing:

the '*' key is also the 'clear input' key so the Reset command must be entered before any other characters in order to be recognized.

To edit text before '#' is pressed, use the '*' key as a clear key to erase data and retype. Press '#' when finished. Data cannot be edited after '#' is pressed

Menu commands will be the same on the PC except [enter] will be used in place of '#'.

Individual commands will be responsible for echoing characters and managing KB_CNT, PC_CNT, ECHO, and checking KB_MODE.

CALIBRATE:

- Procedure to re-calibrate the scale.
- A second order polynomial will be used to map the voltage read from the A/D to the corresponding weight of the mass in the pan.
- Weight will be calculated in 64ths of grams (or could be calculated in tenths of grams if we decide not to use floating point math).
- The calibration curve will take the form: $W = A*V^2 + B*V + C$. Where W is the calculated weight, V is the voltage read from the A/D, and A, B, and C are calculated by the CALIBRATE routine.
- The weights to be placed on the scale will be predefined, and the number of weights is tentatively set to five.
- A least-squares constraint will be imposed on the algorithm to minimize the error between the collected data points and fitted parabolic-like curve.

Registers used for CALIBRATE

Size depends if using fixed or floating point math.

- **AD0, AD1, AD2, AD3, AD4.** voltages read from the A/D that correspond to the predetermined calibration weights.
- **a, b, c.** coefficients for the polynomial: $W = A*V^2 + B*V + C$
- **WTS.** Number of weights left in the calibration procedure. If using five weights to fit curve, then WTS will be initially 5 and will be decremented after each calibration weight is used.

External routines needed for CALIBRATE

- Weight and display current mass on scale.
- Check stability of scale (is measurement accurate?)
- Read in character
- Write character string to display

Pending issues for CALIBRATE:

- C or assembly?
- Fixed or floating point math?
- If use integer math, then how much precision needs to be retained in weight (grams, tenths of grams, or more)?