****
Proj_1F_PCI_and_T1_interrupt

MULTI PATTERN DISPLAY

Uses switch presses to select any one of six displays. A more ambitious program in which the inevitable complexity is handled by two subroutines (see below) and two interrupt service routines one triggered by a timer and the other by a switch press. One of the subroutines is used to initiate the display and the other to increment it.

IT INTRODUCES

1. The Watch Dog Timer: Part of the Atmega HW that can be programmed to generate a reset if program execution unexpectedly halts for any reason. It will be studied in more detail later on.

2. Interrupts from multiple sources: Combining a timer interrupt to increment the display with a PCI interrupt to select the display and initialise it.

3. Lots of logic; once again the study of most of it is probably best postponed.

4. Subroutines: A subroutine executes a segment of code on behalf of the main routine. Subroutines may be used to make a program more readable especially if the code segment is required several times. Alternatively the subroutine may provide a service like some arithmetic, that may be used by just one or by many programs.

Consider the project subroutine void Inc_Display(char mode). It is called by the statement Inc_Display(switch_press);

void Inc_Display(char mode) defines a variable and gives it the name mode. The initial value of mode is supplied by the statement Inc_Display(switch_press) in the main routine in which switch_press is defined as a char variable.

The subroutine can change the value of mode but these changes will not effect the variable switch_press in the main routine. The term void indicates that the subroutine will not be returning any value to the main routine as might be the case for an arithmetic function for example.

5. Volatile variables: The theory: Every variable used by the main routine is allocated permanent storage space in data memory. Variables only used by the subroutines or ISRs are allocated shared memory space in data memory.

However there may be some variables that are not used by the main routine but that must not be allowed to loose their value or be overwritten by other variables. They also require permanent storage and are defined as volatile.

Any variable in an ISR that must be used else where must also be defined as volatile.