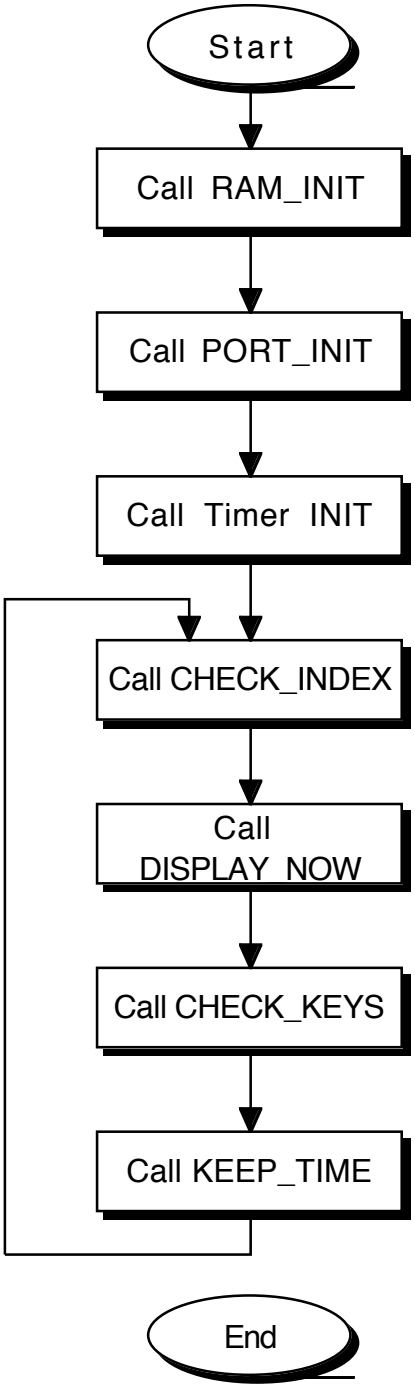
Blick Propeller Clock

"noblink" Main Program Overview

Note: The following is my current understanding of the program.
I guarantee the previous sentence and nothing else contained herein!
0x in front of a number simply means that number is in hex format
i.e. 0x19 is 19 hex (but 25 decimal).



Starts clock at 12:00:00.
Initializes all the variables used in the program to appropriate values.

Sets which ports will be used and what direction they are (input or output).

Enables interrupts.
Sets how crystal clock pulses are counted by the internal timer (1:1 or every clock pulse in this case).
Clears timers.

Checks if "blinker" has just cleared shade. If so it resets digit and dot (column) indexes to initiate a new display sequence and then checks how long it took for a revolution (period_count & period_use) in order to keep a stable display.

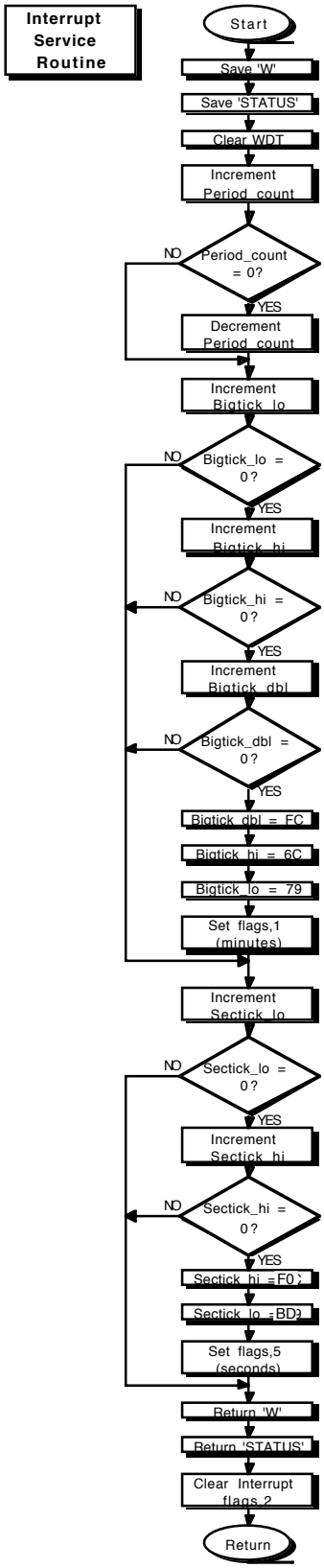
Each CALL displays a single vertical LED column of a single digit only.
The width of the display is based on the value of "period_use".
The display sequence (successive calls) is initiated immediately by the blinker clearing the shade (CHECK_INDEX). The sequence is run until digit & dot (column) indexes have incremented to completion. After that each call basically just returns having displayed nothing until the next reset in CHECK_INDEX.

Checks if someone is trying to set the time with the key switches.

Increments the seconds, minutes and hours based on variables controlled by the INTERRUPT SERVICE ROUTINE or as directed by CHECK_KEYS above.

Interrupt Service Routine

This interrupt routine is run whenever the PIC internal timer completes a count (no matter where in the program to the left it happens to be) and returns when completed. It clears the watchdog timer and keeps track of fractional seconds and sets flags when it has counted out a second or a minute. It also increments the period_count which is used to keep track of a rotational period (used in CHECK_INDEX).



This interrupt routine is run every 256 instruction cycles (256us or 256/1,000,000) when timer overflows from FF --> 00.

Here period_count is incremented. It is not allowed to "reset" if it gets to"FF" i.e. it is held at "FF". Hence, 256us/interrupt x 256 (FF max count) = 65.5msec maximum (or 915.5 RPM is the minimum clock speed). Slower speeds will hit max period_count FF while faster speeds will have a lower period_count (but with progressively lower "resolution").

Here the number of interrupts is counted to keep track of minutes. Every 234375 times this interrupt routine is run the minute flag (bit 1) is set.

i.e. 256us x 234375 = 60 sec (1 min)

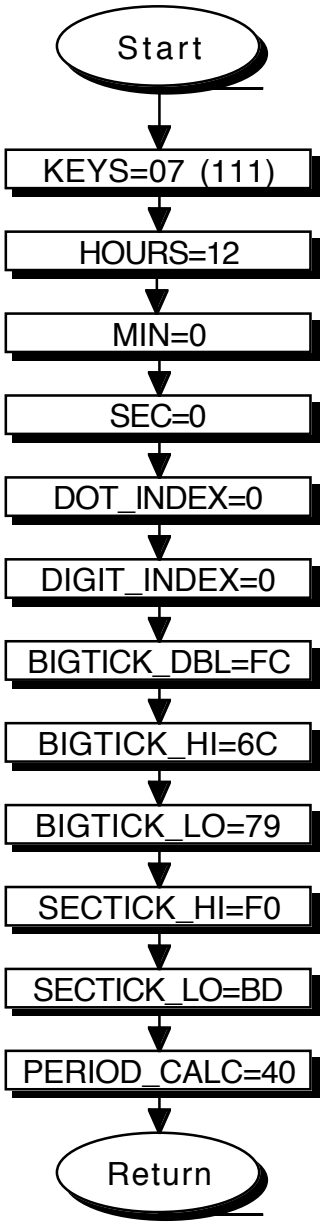
234375 = 0x039387
0x000000 - 0x039387 = 0xFC6C79 (the number to count up from). Therefore bigtick_dbl is set to FC, bigtick_hi is set to 6C while bigtick_lo is set to 79 and the interrupt routine increments from this value (up to 0).

Here the number of interrupts is counted to keep track of seconds. Every 3907 times this interrupt routine is run the seconds flag (bit 5) is set.

i.e. 256us x 3907 = 1.000192 sec

3907 = 0x0F43
0x0000 - 0x0F43 = 0xF0BD (the number to count up from). Therefore sectick_hi is set to F0 while sectick_lo is set to BD and the interrupt routine increments from this value (up to 0).

RAM_INIT Subroutine



Reset the time setting keys

Start the clock at 12:00:00

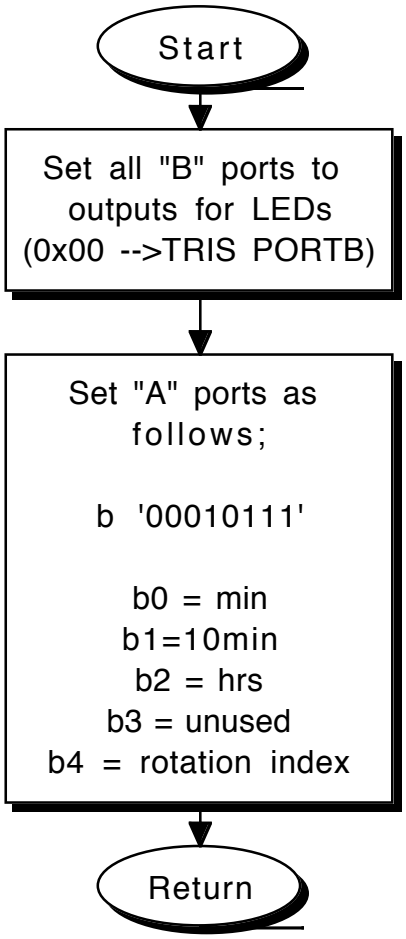
Reset the digit and column index

Reset the 3 values that are used to count out a minute

Reset the 2 values that are used to count out a second

Reset the rotation time value

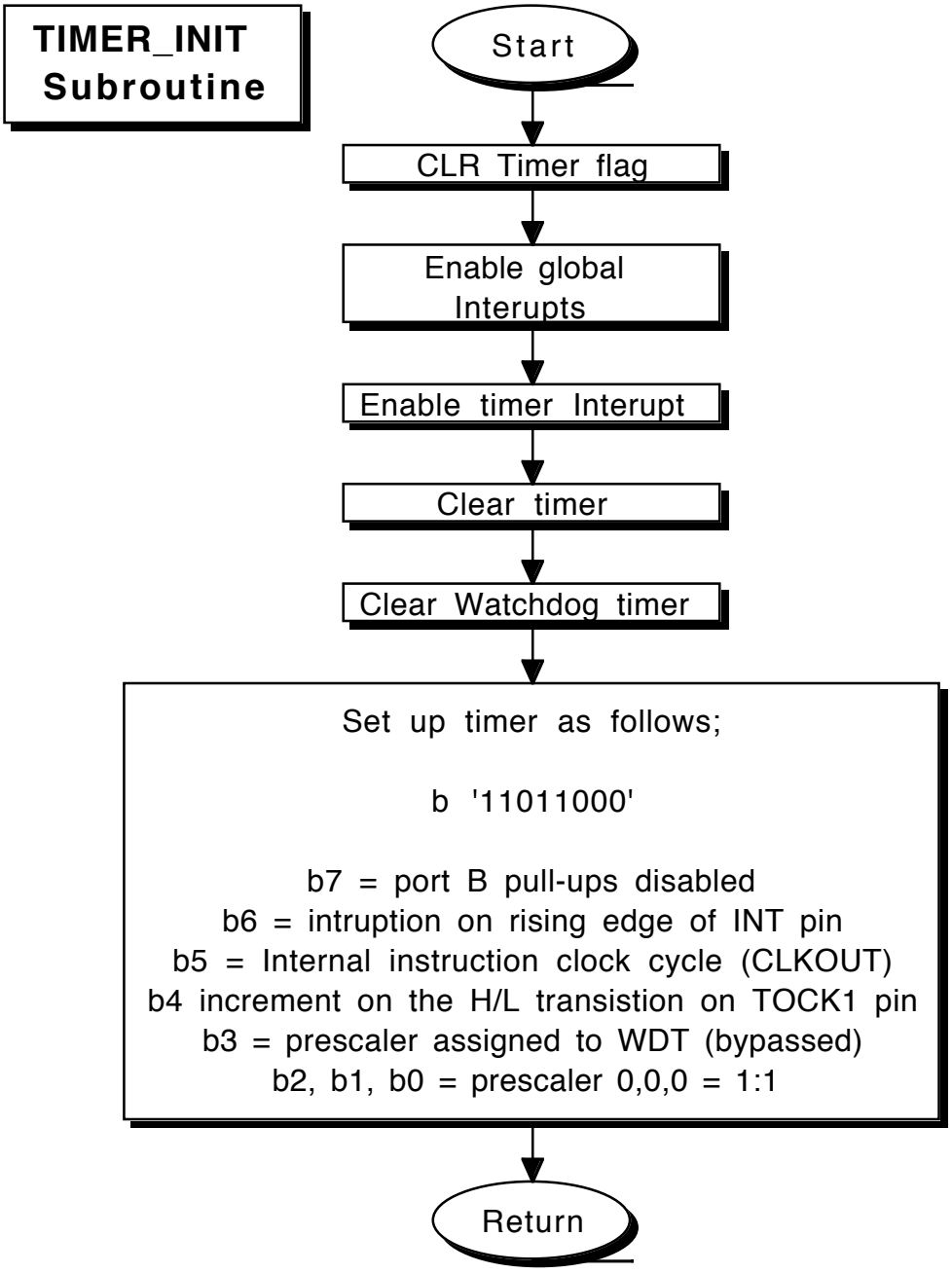
PORT_INIT Subroutine

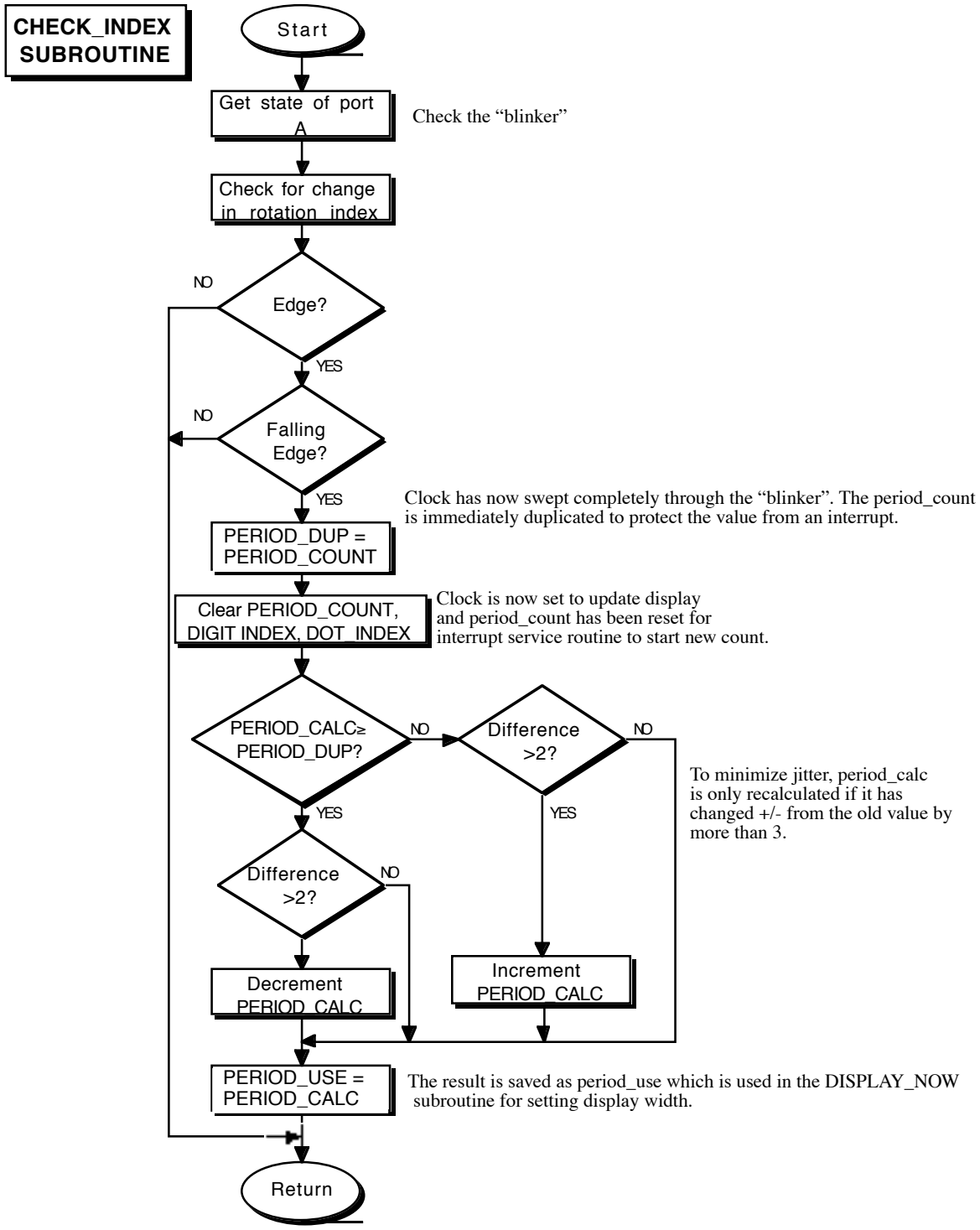


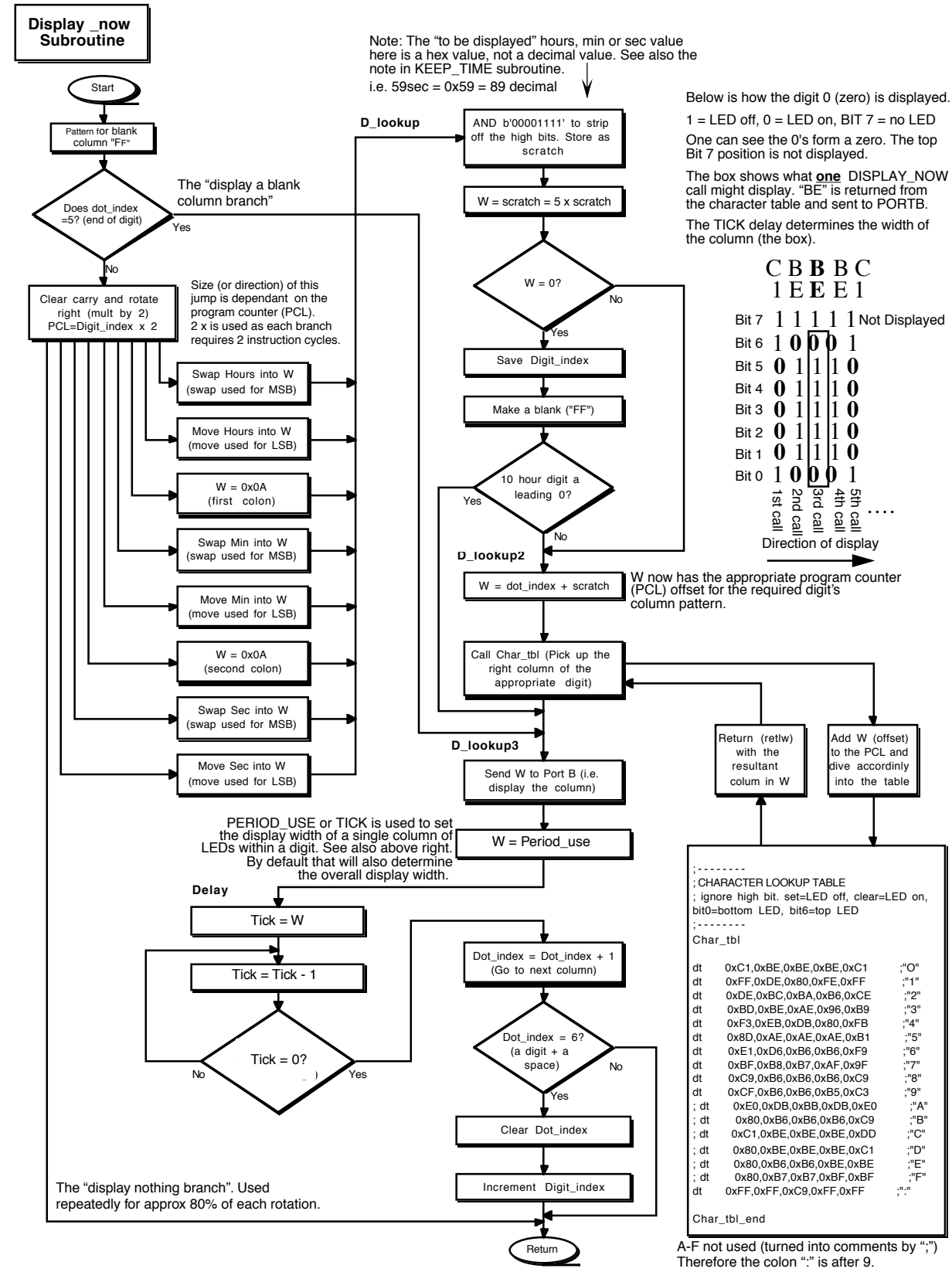
A “0” sets a port to an output while a “1” sets it to an input.

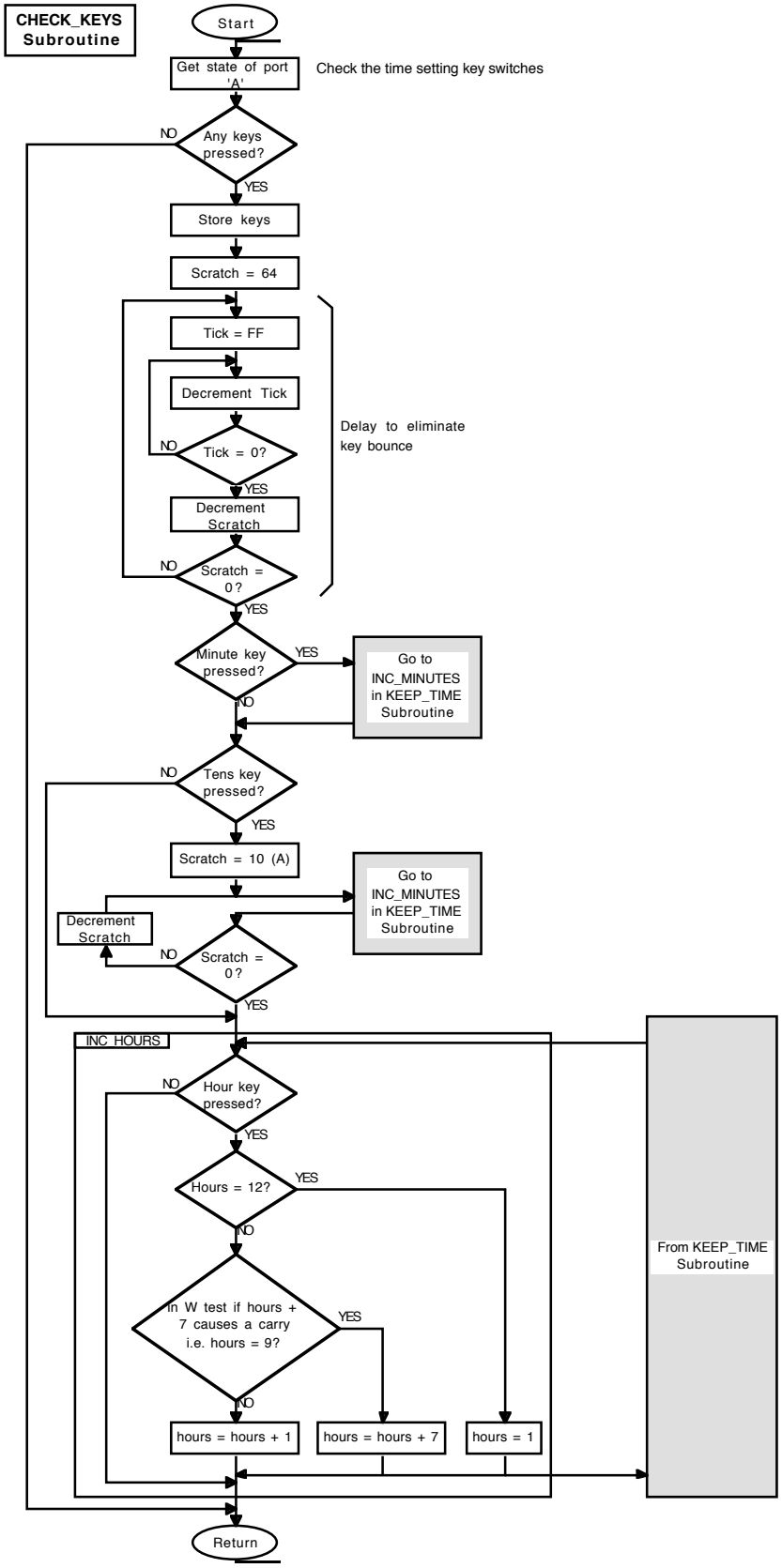
b0-b2 for time-setting key switch input.

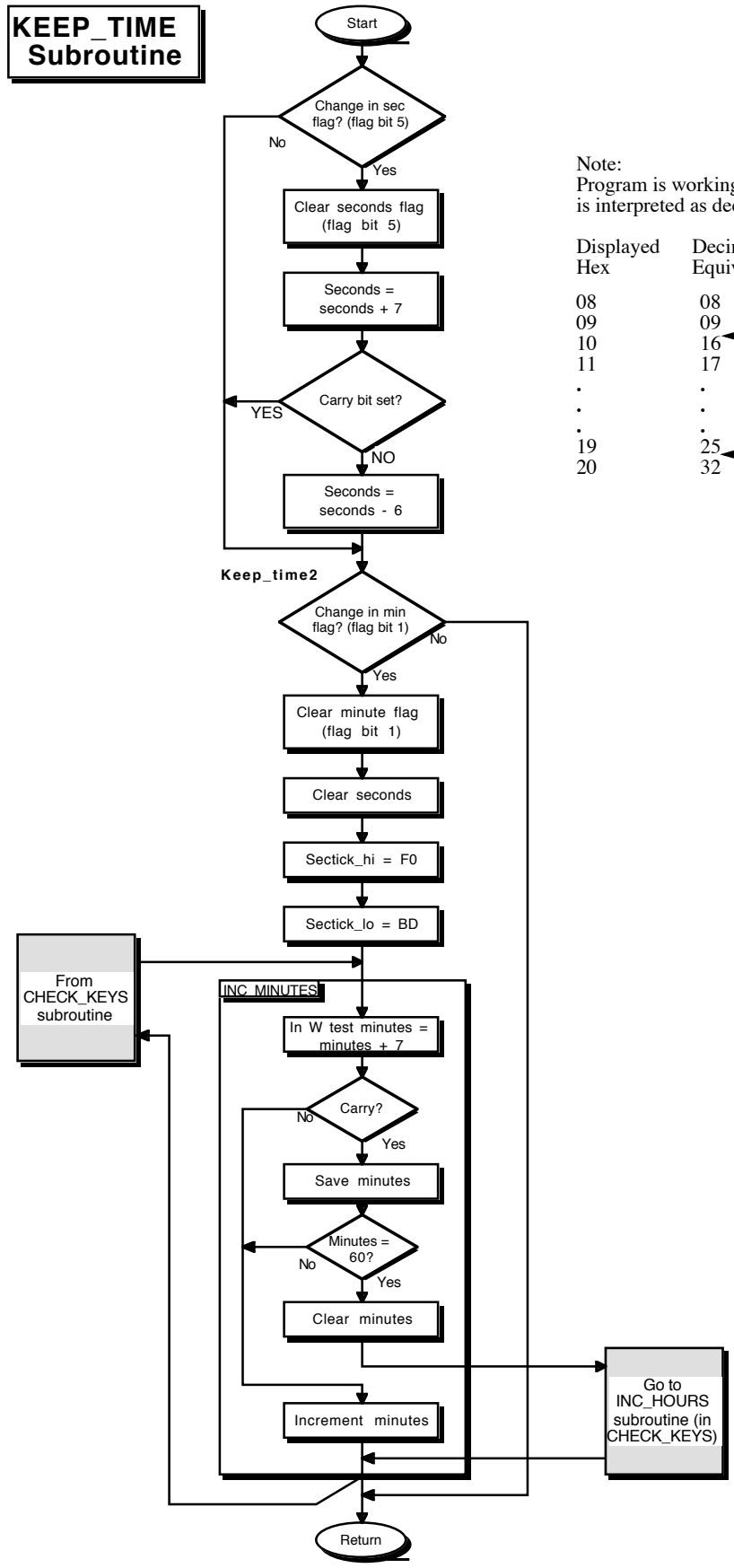
b4 for “blinker” input


TIMER_INIT
Subroutine
Start
CLR Timer flag
Enable global
Interupts
Enable timer Interupt
Clear timer
Clear Watchdog timer
Set up timer as follows;
b '11011000'
b7 = port B pull-ups disabled
b6 = intruption on rising edge of INT pin
b5 = Internal instruction clock cycle (CLKOUT)
b4 increment on the H/L transistion on TOCK1 pin
b3 = prescaler assigned to WDT (bypassed)
b2, b1, b0 = prescaler 0,0,0 = 1:1
Return


CHECK_INDEX SUBROUTINE
Start
Get state of port A
Check the "blinker"
Check for change in rotation index
NO
Edge?
YES
NO
Falling Edge?
YES
Clock has now swept completely through the "blinker". The period_count is immediately duplicated to protect the value from an interrupt.
PERIOD_DUP = PERIOD_COUNT
Clear PERIOD_COUNT, DIGIT INDEX, DOT_INDEX
Clock is now set to update display and period_count has been reset for interrupt service routine to start new count.
PERIOD_CALC≥ PERIOD_DUP?
NO
Difference >2?
NO
YES
YES
To minimize jitter, period_calc is only recalculated if it has changed +/- from the old value by more than 3.
Difference >2?
NO
Decrement PERIOD_CALC
Increment PERIOD_CALC
PERIOD_USE = PERIOD_CALC
The result is saved as period_use which is used in the DISPLAY_NOW subroutine for setting display width.
Return


Display _now Subroutine
Start
Pattern for blank column "FF"
Does dot_index =5? (end of digit)
Yes
No
The "display a blank column branch"
Clear carry and rotate right (mult by 2) PCL=Digit_index x 2
Size (or direction) of this jump is dependant on the program counter (PCL). 2 x is used as each branch requires 2 instruction cycles.
Swap Hours into W (swap used for MSB)
Move Hours into W (move used for LSB)
W = 0x0A (first colon)
Swap Min into W (swap used for MSB)
Move Min into W (move used for LSB)
W = 0x0A (second colon)
Swap Sec into W (swap used for MSB)
Move Sec into W (move used for LSB)
The "display nothing branch". Used repeatedly for approx 80% of each rotation.
Note: The "to be displayed" hours, min or sec value here is a hex value, not a decimal value. See also the note in KEEP_TIME subroutine.
i.e. 59sec = 0x59 = 89 decimal
D_lookup
AND b'00001111' to strip off the high bits. Store as scratch
W = scratch = 5 x scratch
W = 0?
No
Yes
Save Digit_index
Make a blank ("FF")
10 hour digit a leading 0?
Yes
No
D_lookup2
W = dot_index + scratch
W now has the appropriate program counter (PCL) offset for the required digit's column pattern.
Call Char_tbl (Pick up the right column of the appropriate digit)
D_lookup3
Send W to Port B (i.e. display the column)
PERIOD_USE or TICK is used to set the display width of a single column of LEDs within a digit. See also above right. By default that will also determine the overall display width.
W = Period_use
Delay
Tick = W
Tick = Tick - 1
Tick = 0?
No
Yes
Dot_index = Dot_index + 1 (Go to next column)
Dot_index = 6? (a digit + a space)
No
Yes
Clear Dot_index
Increment Digit_index
Return
Return (retlw) with the resultant colum in W
Add W (offset) to the PCL and dive accordinly into the table
;--------
; CHARACTER LOOKUP TABLE
; ignore high bit. set=LED off, clear=LED on, bit0=bottom LED, bit6=top LED
;--------
Char_tbl
dt 0xC1,0xBE,0xBE,0xBE,0xC1 ;"O"
dt 0xFF,0xDE,0x80,0xFE,0xFF ;"1"
dt 0xDE,0xBC,0xBA,0xB6,0xCE ;"2"
dt 0xBD,0xBE,0xAE,0x96,0xB9 ;"3"
dt 0xF3,0xEB,0xDB,0x80,0xFB ;"4"
dt 0x8D,0xAE,0xAE,0xAE,0xB1 ;"5"
dt 0xE1,0xD6,0xB6,0xB6,0xF9 ;"6"
dt 0xBF,0xB8,0xB7,0xAF,0x9F ;"7"
dt 0xC9,0xB6,0xB6,0xB6,0xC9 ;"8"
dt 0xCF,0xB6,0xB6,0xB5,0xC3 ;"9"
; dt 0xE0,0xDB,0xBB,0xDB,0xE0 ;"A"
; dt 0x80,0xB6,0xB6,0xB6,0xC9 ;"B"
; dt 0xC1,0xBE,0xBE,0xBE,0xDD ;"C"
; dt 0x80,0xBE,0xBE,0xBE,0xC1 ;"D"
; dt 0x80,0xB6,0xB6,0xBE,0xBE ;"E"
; dt 0x80,0xB7,0xB7,0xBF,0xBF ;"F"
dt 0xFF,0xFF,0xC9,0xFF,0xFF ;":"
Char_tbl_end
A-F not used (turned into comments by ";")
Therefore the colon ":" is after 9.
Below is how the digit 0 (zero) is displayed.
1 = LED off, 0 = LED on, BIT 7 = no LED
One can see the 0's form a zero. The top Bit 7 position is not displayed.
The box shows what one DISPLAY_NOW call might display. "BE" is returned from the character table and sent to PORTB.
The TICK delay determines the width of the column (the box).
C B B B C
1 E E E 1
Bit 7 1 1 1 1 1 Not Displayed
Bit 6 1 0 0 0 1
Bit 5 0 1 1 1 0
Bit 4 0 1 1 1 0
Bit 3 0 1 1 1 0
Bit 2 0 1 1 1 0
Bit 1 0 1 1 1 0
Bit 0 1 0 0 0 1
1st call
2nd call
3rd call
4th call
5th call
....
Direction of display


CHECK_KEYS Subroutine
Start
Get state of port 'A'
Check the time setting key switches
Any keys pressed?
NO
YES
Store keys
Scratch = 64
Tick = FF
Decrement Tick
Tick = 0?
NO
YES
Decrement Scratch
Scratch = 0?
NO
YES
Delay to eliminate key bounce
Minute key pressed?
YES
NO
Go to INC_MINUTES in KEEP_TIME Subroutine
Tens key pressed?
NO
YES
Scratch = 10 (A)
Go to INC_MINUTES in KEEP_TIME Subroutine
Decrement Scratch
Scratch = 0?
NO
YES
INC_HOURS
Hour key pressed?
NO
YES
Hours = 12?
YES
NO
In W test if hours + 7 causes a carry i.e. hours = 9?
YES
NO
hours = hours + 1
hours = hours + 7
hours = 1
From KEEP_TIME Subroutine
Return



Note:
Program is working in Hex but display is interpreted as decimal hence jumps of 7.

Displayed Hex	Decimal Equivalent	
08	08	
09	09	Jump of 7
10	16	
11	17	
.	.	
.	.	
.	.	
19	25	Jump of 7
20	32	