DELAY FUNCTIONS

The delay functions execute code for a specific number of processor instruction cycles. For time based delays, the processor operating frequency must be taken into account. The following routines are provided:

TABLE 4-4: DELAY FUNCTIONS

Function	Description
Delay1TCY	Delay one instruction cycle.
Delay10TCYx	Delay in multiples of 10 instruction cycles.
Delay100TCYx	Delay in multiples of 100 instruction cycles.
Delay1KTCYx	Delay in multiples of 1,000 instruction cycles.
Delay10KTCYx	Delay in multiples of 10,000 instruction cycles.

4.5.1 Function Descriptions

Delay1TCY

Function: Delay 1 instruction cycle (TCY).

Include: delays.h

Prototype: `void Delay1TCY( void );`

Remarks: This function is actually a `#define` for the `NOP` instruction. When encountered in the source code, the compiler simply inserts a `NOP`.

File Name: `#define in delays.h`

Delay10TCYx

Function: Delay in multiples of 10 instruction cycles (TCY).

Include: delays.h

Prototype: `void Delay10TCYx( unsigned char unit );`

Arguments: ***unit***
The value of ***unit*** can be any 8-bit value. A value in the range [1,255] will delay (***unit*** * 10) cycles. A value of 0 causes a delay of 2,560 cycles.

Remarks: This function creates a delay in multiples of 10 instruction cycles.

File Name: d10tcyx.asm

Delay100TCYx

Function: Delay in multiples of 100 instruction cycles (TCY).

Include: delays.h

Prototype: `void Delay100TCYx( unsigned char unit );`

Arguments: ***unit***
The value of ***unit*** can be any 8-bit value. A value in the range [1,255] will delay (***unit*** * 100) cycles. A value of 0 causes a delay of 25,600 cycles.

Delay100TCYx (Continued)

Remarks: This function creates a delay in multiples of 100 instruction cycles. This function uses the globally allocated variable, `DelayCounter1`. If this function is used in both interrupt and mainline code, the variable `DelayCounter1` should be saved and restored in the interrupt handler. Refer to the `save=` clause of the `#pragma interrupt` or `#pragma interruptlow` directives for more information. Note that other delay functions also use the globally allocated `DelayCounter1` variable.

File Name: `d100tcyx.asm`

Delay1KTCYx

Function: Delay in multiples of 1,000 instruction cycles (TCY).

Include: `delays.h`

Prototype: `void Delay1KTCYx( unsigned char unit );`

Arguments: ***unit***
The value of ***unit*** can be any 8-bit value. A value in the range [1,255] will delay (***unit*** * 1000) cycles. A value of 0 causes a delay of 256,000 cycles.

Remarks: This function creates a delay in multiples of 1,000 instruction cycles. This function uses the globally allocated variables, `DelayCounter1` and `DelayCounter2`. If this function is used in both interrupt and mainline code, these variables, `DelayCounter1` and `DelayCounter2`, should be saved and restored in the interrupt handler. Refer to the `save=` clause of the `#pragma interrupt` and `#pragma interruptlow` directives for more information. Note that other delay functions also use the globally allocated `DelayCounter1` variable.

File Name: `d1ktcyx.asm`

Delay10KTCYx

Function: Delay in multiples of 10,000 instruction cycles (TCY).

Include: `delays.h`

Prototype: `void Delay10KTCYx( unsigned char unit );`

Arguments: ***unit***
The value of ***unit*** can be any 8-bit value. A value in the range [1,255] will delay (***unit*** * 10000) cycles. A value of 0 causes a delay of 2,560,000 cycles.

Remarks: This function creates a delay in multiples of 10,000 instruction cycles. This function uses the globally allocated variable, `DelayCounter1`. If this function is used in both interrupt and mainline code, the variable `DelayCounter1` should be saved and restored in the interrupt handler. Refer to the `save=` clause of the `#pragma interrupt` or `#pragma interruptlow` directives for more information. Note that other delay functions also use the globally allocated `DelayCounter1` variable.

File Name: `d10ktcyx.asm`

Delays functions use:

#include <delays.h>

void Delay1TCYx (void); //Delay one instruction cycle
void Delay 10TCYx(unsigned char ***unit***); //Delay in multples of 10 instruction cycles
void Delay 100TCYx (unsigned char ***unit***); //Delay in multples of 100 instruction cycles
void Delay1KTCYx (unsigned char ***unit***); //Delay in multples of 1000 instruction cycles
void Delay10KTCYx (unsigned char ***unit***); //Delay in multples of 10,000 instruction cycles

The letter 'x' in the function name specifies times ie multiplication .

The **unit** has an 8-bit value (0-255). A **unit** value of 0 is equivalent to unit=256

The following examples will make this explanation clear:

TCY stands for ***Instruction Cycle.*** Our PIC18F4520 is clocked by an external HS clock of 20MHz. This clock is internally divided by 4.
Therefore TCY= 4/20MHz = 200ns (time taken to execute one nop single cycle instruction)

Examples

Delay1TCY() ; //gives a delay of 0,2us = 200ns

Delay1TCYx(100); //gives a delay of 100 x 4/20 = 20us

Delay1KTCYx(100); // gives a delay of 1000 x 100 x 4/20 = 20ms

Delay 10KTCYx(100); // gives a delay of 10,000 x100 x 4/20 = 0,2 S

Delay10KTCYx(0) //gives a delay of 10,000 x 256 x 4/20 = 0.5 s