Background:

The clock comes on and stabilizes. The first thing to do is set it up. Here are the basics. Later the program flow is described.

> 5.2 After a PUC, MCLK and SMCLK are sourced from DCOCLK at 32 times the ACLK frequency. When a **32 768-Hz** crystal is used for **ACLK**, **MCLK** and **SMCLK** stabilize to **1.048576 Mhz**.

Get the MCLK, ACLK, and SMCLK running at max speed, 12.288MHz

1. Configure XT2 hardware
FLL_CTL2
Set XT2Sx, bits 7-6, as 11b for .4 to 16MHz input
FLL_CTL1
Clear bit 5 to make sure XT2OFF is not set

2. Connect the clocks MCLK and SMCLK. ACLK remains 32.768KHz, probably unused.
FLL_CTL1
Set bits 4-3 to 10b for XT2CLK (CPUOFF should be clear)
Set bit 2 for smclk source to XT2CLK
Clear bit 6 to keep SMCLK on

(are globals enabled? Does it matter? Answer this question here when found)
3. Clear faults
IE1: Bit 1, OFIE, enables interrupt. (may flag perhipheral interrupt spuriously)
IFG1: Read bit 1, OFIFG, to detect interrupt pending.

Program flow:

The low hanging fruit is to connect the clocks. Do steps 1 and 2 in described order, configure then enable.

The harder task is to clear the faults when the appear. It would be good to report clock fault failures with a **global** variable. Use the interrupt to clear the flag then wait a set time, 100ms ish. Count the failures.

First enable the interrupts with by setting OFIE in IE1. Most presumably the interrupt will fire immediately. When the PC comes back from the interrupt, code execution will begin.

Files:
The C file shall contain a initialization routine, an interrupt service routine. The ISR turns off the LFOSC, makes sure the clocks are connected, and clears the flag. What else can it do?