## ****MCU Clock System****

**وظيفته:** يعطي تردد ثابت (Clock) لكل أجزاء الميكروكنترولر ليشتغل بتناغم.

**مكونات أساسية:**

**Internal Oscillator:** مولد تردد داخلي، سريع وسهل الاستخدام لكنه أقل دقة.

**External Crystal / Resonator:** دقيق أكثر، يستخدم للتطبيقات اللي تحتاج توقيت ثابت.

**PLL (Phase-Locked Loop):** يضاعف تردد المصدر للحصول على سرعة أعلى.

**ملاحظات:**

Clock يؤثر مباشرة على سرعة المعالجة، زمن تنفيذ التعليمات، ودقة الوقت في الـ Timers.

## ****Interrupt Fundamentals & Architecture****

**Interrupt:** آلية لإيقاف البرنامج الرئيسي مؤقتًا لمعالجة حدث مهم (مثل زر أو حساس).

**أنواع Interrupts:**

**External Interrupts:** من الأجهزة الخارجية (زر، حساس).

**Internal Interrupts:** من الأجهزة الداخلية (Timers, ADC).

**Architecture:**

كل MCU عنده **Interrupt Vector Table**: جدول يحدد أي ISR (Interrupt Service Routine) يتنفذ لأي حدث.

أولوية المعالجة (Priority) تحدد أي مقاطعة تتنفذ أولًا عند حدوث أكثر من واحدة.

## ****Interrupt Handling & Startup Process****

**Handling Steps:**

حدث Interrupt يحدث.

MCU يحفظ حالة المعالج (Registers) مؤقتًا.

يقف البرنامج الرئيسي ويقفز إلى ISR المناسب من الـ Vector Table.

ينفذ ISR.

يعيد الحالة السابقة ويكمل البرنامج الرئيسي.

**Startup Process:**

بعد تشغيل MCU:

Reset → Initialize Stack & Registers → Configure Clock → Enable Interrupts → Jump to main program.

## ****Software Mechanisms for Managing Interrupts****

**ISR (Interrupt Service Routine):** الدالة اللي تتنفذ عند حدوث الـ Interrupt.

**Enable / Disable Interrupts:**

يمكن تعطيل المقاطعات مؤقتًا لمنع التعارض أثناء تحديث متغيرات حساسة.

**Flags & Registers:**

كل Interrupt له Flag في Register يوضح هل حصل الحدث.

ISR غالبًا يمسح الـ Flag بعد المعالجة.

**Nested Interrupts:**

بعض MCUs تدعم مقاطعة ISR أثناء تنفيذ ISR آخر، مع مراعاة الأولويات.

**Best Practices:**

اجعل ISR قصيرة وسريعة.

تجنب العمليات الثقيلة داخل ISR.

استخدم Flags للتعامل مع المعالجة الكبيرة خارج ISR في البرنامج الرئيسي.