विषय सूची

अध्याय		पृष्ठ संख्या
1.	माइक्रोकंट्रोलर सीरीज (Micro-controller Series (MCS)	1-27
2.	माइक्रोकंट्रोलर प्रोग्रामिंग के लिए निर्देश सेट (Instruction Set for Micro-controller Programming)	28-52
3.	एम्बेडेड सिस्टम का परिचय (Introduction to Embedded System)	53-70
4.	एम्बेडेड ऑपरेशन सिस्टम (Embedded Operating Systems)	71-90
5.	पीआईसी माइक्रोकंट्रोलर का परिचय (Introduction of PIC Microcontroller)	91-101
6.	माइक्रोकंट्रोलर के प्रोग्रामिंग कॉन्सेप्टस (Programming Concepts of Microcontroller)	102-131
7.	इनपुट/आउटपुट इंटरफ़ेस (Input/output Interface)	132-149
8.	इंटरनेट ऑफ थिंग्स (Internet of Things)	150-166
●	प्रयोगात्मक कार्य (Practicals)	167-199

माइक्रोकंट्रोलर सीरीज

(Micro-controller Series (MCS)

SYLLABUS

- Architecture of 8051 Microcontroller
- Pin details
- I/O Port structure
- Memory Organization
- Special Function
- External Memory

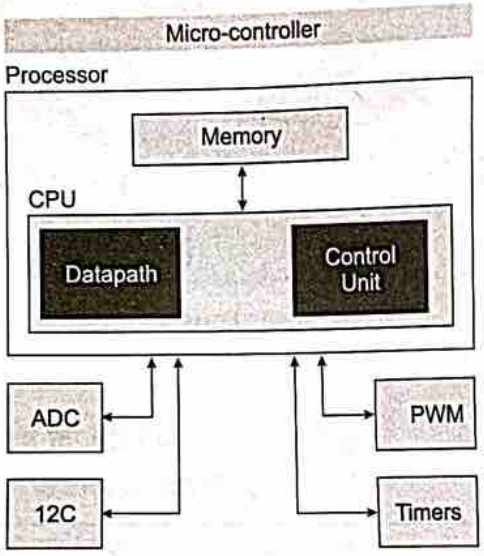
1.1 परिचय (Introduction)

आधुनिक जीवन को आकार देने वाली तकनीकी क्रांति में माइक्रोकंट्रोलर ने प्रमुख भूमिका निभाई है। माइक्रोकंट्रोलर छोटे, versatile, व सस्ते उपकरण हैं, जिन्हें न केवल अनुभवी इलैक्ट्रिकल इंजीनियरों द्वारा, बल्कि अन्य विषयों में दक्ष, छात्रों और पेशेवरों द्वारा भी सफलतापूर्वक कार्यान्वित और प्रोग्राम किया जा सकता है। संभावित माइक्रोकंट्रोलर अनुप्रयोगों की सूची बहुत लंबी है। कम लागत वाले उपकरण, चिकित्सा उपकरण, उच्च अंत उपभोक्ता इलैक्ट्रॉनिक्स, औद्योगिक उपकरण, अत्याधुनिक सैन्य और एयरोस्पेस सिस्टम (Low-cost wearables, medical equipment, high-end consumer electronics, rugged industrial devices, state-of-the-art military and aerospace systems) अनुकूलनीय, सस्ते, उपयोगकर्ता के अनुकूल घटक किसी भी इलैक्ट्रॉनिक उत्पाद के बहुत अच्छे उदाहरण हैं।

माइक्रोकंट्रोलर एक Integrated circuit (IC) उपकरण है, जिसका उपयोग इलैक्ट्रॉनिक सिस्टम के अन्य भागों को नियंत्रित करने के लिए किया जाता है। इन उपकरणों को एम्बेडेड अनुप्रयोगों के लिए बनाया जाता है, जिन्हें तीव्र कार्यक्षमता के लिए डिजिटल, एनालॉग या इलैक्ट्रोमैकेनिकल घटकों के उपयोग से बनाया जाता है। Integrated circuit की इस श्रेणी को परिभाषित करने का सबसे सरल तरीका 'माइक्रोकंट्रोलर' है, लेकिन संक्षिप्त रूप से इसके लिए 'MCU' का उपयोग किया जाता है क्योंकि यह "माइक्रोकंट्रोलर यूनिट" के लिए प्रयोग होता है। इसको कभी-कभी 'μC' (ग्रीक अक्षर μ) भी कहते हैं।'

1.2 माइक्रोकंट्रोलर vs. माइक्रोप्रोसेसर (Micro-controllers vs. Micro-processors)

लोग कभी-कभी 'माइक्रोप्रोसेसर' या 'MPU' शब्द का उपयोग करते हैं, लेकिन ये दोनों डिवाइस समान नहीं हैं। माइक्रोप्रोसेसर और माइक्रोकंट्रोलर दोनों छोटे, उच्च एकीकृत कंप्यूटर सिस्टम के रूप में कार्य करते हैं, लेकिन वे विभिन्न उद्देश्यों की पूर्ति करते हैं। शब्द 'प्रोसेसर' का उपयोग एक ऐसी प्रणाली की पहचान करने के लिए किया जाता है, जिसमें एक सेंट्रल प्रोसेसिंग यूनिट होती है और (वैकल्पिक रूप से) कुछ मेमोरी। माइक्रोप्रोसेसर एक ऐसा उपकरण है, जो प्रोसेसर के सभी कार्यों को एक इंटीग्रेटेड सर्किट के अन्दर संरक्षित करता है, जबकि Micro-controller, अतिरिक्त हार्डवेयर मॉड्यूल हैं, जो उपकरण को केवल इंस्ट्रक्शन को निष्पादित करने और डाटा को स्टोर करने के अतिरिक्त सिस्टम को नियंत्रित करते हैं।



चित्र 1.1

यद्यपि जब हम एक शब्द का बार-बार उपयोग करने से बचना चाहते हैं, तो हम 'माइक्रोप्रोसेसर' और 'माइक्रोकंट्रोलर' शब्दों का प्रयोग कर सकते हैं। हालाँकि, तकनीकी चर्चा के संदर्भ में, दो अवधारणाओं के बीच अंतर बनाए रखना महत्वपूर्ण है।

माइक्रोप्रोसेसर और माइक्रोकंट्रोलर के बीच के अंतर को दी गई तालिका में दिया गया है—

	Micro-controller	Micro-processor
1.	माइक्रोकंट्रोलर का उपयोग किसी एप्लिकेशन में किसी एक कार्य को करने के लिए किया जाता है।	माइक्रोप्रोसेसर का उपयोग बड़े एप्लिकेशन के लिए किया जाता है।
2.	इसकी डिजाइनिंग और हार्डवेयर का मूल्य कम है।	इसकी डिजाइनिंग और हार्डवेयर का मूल्य अधिक है।
3.	इसको रिप्लेस करना आसान है।	इसको रिप्लेस करना आसान नही है
4.	इसे CMOS तकनीक से बनाया गया है, जिसे संचालित करने के लिए कम शक्ति की आवश्यकता होती है।	इसकी बिजली की खपत अधिक है, क्योंकि यह पूरी प्रणाली को नियंत्रित करता है।
5.	इसमें CPU, RAM, ROM, I/O पोर्ट होते हैं।	इसमें RAM, ROM, I/O पोर्ट नहीं होते। इसकी पिन का उपयोग पेरिफेरेल उपकरणों के इंटरफेस के लिए करते हैं।

1.3 माइक्रोकंट्रोलर के घटक (The Elements of a Microcontroller)

माइक्रोकंट्रोलर में एक सेंट्रल प्रोसेसिंग यूनिट (सीपीयू), Non-volatile, Volatile Memory, Memory, पेरिफेरल और सपोर्ट परिपथ होते हैं, जिनके बारे में विस्तारपूर्वक यहाँ वर्णित किया गया है।

एक माइक्रोकंट्रोलर के मुख्य तत्व निम्नलिखित हैं—

प्रोसेसर (CPU)—प्रोसेसर को डिवाइस का मस्तिष्क कहते हैं। यह माइक्रोकंट्रोलर के कार्य को निर्देशित करने वाले उन विभिन्न निर्देशों पर क्रिया और प्रतिक्रिया करता है जो बेसिक एरियमेटिक, लॉजिकल और I/O संचालन करते हैं। यह डाटा ट्रांसफर संचालन भी करता है, जो बड़े एम्बेडेड सिस्टम में अन्य घटकों में कमांड का संचार करते है।

मेमोरी (Memory)—माइक्रोकंट्रोलर की मेमोरी का उपयोग डाटा को संग्रहीत करने के लिए किया जाता है, जो प्रोसेसर प्राप्त करता है और उन निर्देशों का उत्तर देने के लिए भी उपयोग किया जाता है, जिसे इसको करने के लिए प्रोग्राम किया गया है। एक माइक्रोकंट्रोलर में दो मुख्य मेमोरी प्रकार होती हैं।

1. प्रोग्राम मेमोरी (Program Memory)—यह सीपीयू द्वारा दिए गए निर्देशों को लम्बे समय तक संग्रहीत करता है। प्रोग्राम मेमोरी नॉन-वोलेटाइल मेमोरी है, जिसका अर्थ है, कि यह एक शक्ति स्रोत की के बिना समय के साथ जानकारी रखता है।

2. डाटा मेमोरी (Data Memory)—यह मेमोरी डाटा के अस्थायी संग्रहण के लिए आवश्यक होती है, डाटा मेमोरी वोलेटाइल है, जिसका अर्थ है, कि इसमें जो डाटा है वो अस्थायी है और केवल तभी रखा जाता है जब डिवाइस एक शक्ति स्रोत से जुड़ा होता है ।

I/O पेरिफेरेल्स (I/O Peripherals)—इनपुट और आउटपुट डिवाइस बाह्य प्रोसेसर के लिए इंटरफ़ेस होते हैं। इनपुट पोर्ट जानकारी प्राप्त करते हैं और इसे बाइनरी डाटा के रूप में प्रोसेसर को भेजते हैं। प्रोसेसर उस डाटा को प्राप्त करता है और आउटपुट डिवाइसों को आवश्यक निर्देश भेजता है, जो कि माइक्रोकंट्रोलर के बाह्य कार्यों को निष्पादित करता है।

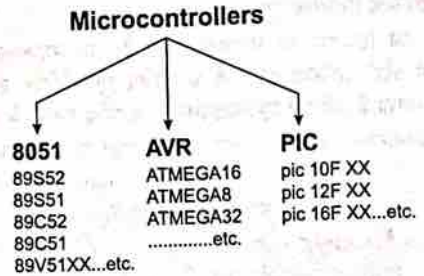
जबकि प्रोसेसर, मेमोरी और I/O माइक्रोप्रोसेसर के बाह्य उपकरणों के परिभाषित तत्व हैं, इसके अन्य भी तत्व हैं। I/O केवल सहायक घटकों के बाह्य उपकरण हैं जो मेमोरी और प्रोसेसर के साथ इंटरफेस करते हैं। इसमें कई सहायक घटक हैं, जिन्हें बाह्य उपकरणों के रूप में वर्गीकृत किया जा सकता है।

माइक्रोकंट्रोलर के अन्य सहायक तत्व निम्नलिखित हैं—

- **एनालॉग से डिजिटल कनवर्टर** (Analog to Digital Converter) (ADC)—ADC एक ऐसा परिपथ है जो एनालॉग सिग्नल को डिजिटल सिग्नल में बदलता है। यह प्रोसेसर को माइक्रोकंट्रोलर के मध्य में बाहरी एनालॉग उपकरणों के साथ इंटरफेस करने की अनुमति देता है, जैसे सेंसर।
- **डिजिटल से एनालॉग कनवर्टर** (Digital to Analog Converter) (DAC)—एक ADC के विपरीत कार्य करता है और यह यह प्रोसेसर को माइक्रोकंट्रोलर के सेंटर में बाह्य एनालॉग घटकों के साथ कम्यूनिकेट करने की अनुमति देता है और डिजिटल सिग्नल को एनालॉग में बदलता है।
- **सिस्टम बस** (System Bus)—सिस्टम बस Connective wire तार है, जो माइक्रोकंट्रोलर के सभी घटकों को एक साथ जोड़ता है ।
- **सीरियल पोर्ट** (Serial Port)—सीरियल पोर्ट I/O पोर्ट का एक उदाहरण है, जो माइक्रोकंट्रोलर को बाह्य घटकों से संयोजित करता है। यह एक USB या समानांतर पोर्ट के समान कार्य करता है, लेकिन यह बिट्स के आदान-प्रदान के तरीके में भिन्न होता है।

1.4 माइक्रोकंट्रोलर के प्रकार (Types of Micro-controllers)

माइक्रोकंट्रोलर के प्रकार दिए गए डायग्राम में दिखाए गए हैं, वे अपने बिट्स, मेमोरी आर्किटेक्चर, मेमोरी/डिवाइसेस और इंस्ट्रक्शन सेट की विशेषता रखते हैं। आइए इनके बारे में संक्षेप में चर्चा करते हैं।



चित्र 1.2

1.4.1 बिट्स की संख्या के आधार पर माइक्रोकंट्रोलर के प्रकार (Types of Microcontrollers Based on the Number of Bits)

माइक्रोकंट्रोलर में 8-बिट्स, 16-बिट्स और 32-बिट्स माइक्रोकंट्रोलर होते हैं।

8-बिट माइक्रोकंट्रोलर में, जब आंतरिक बस 8-बिट होती हैं, तब ALU Arithmetic और Logical संचालन करता है। 8-बिट माइक्रोकंट्रोलर के उदाहरण, इंटेल 8031/8051, PIC1x, और मोटोरोला MC68HC11 हैं।

8-बिट की तुलना में 16-बिट माइक्रोकंट्रोलर अधिक शुद्ध प्रदर्शन करता है। उदाहरण के लिए, 8-बिट माइक्रोकंट्रोलर केवल 8 बिट्स का उपयोग कर सकते हैं, जिसके परिणामस्वरूप हर चक्र के लिए 0 × 00-0xFF (0-255) की अंतिम सीमा होती है। इसके विपरीत, अपनी बिट डाटा विड्थ के साथ 16-बिट माइक्रोकंट्रोलर्स में प्रत्येक चक्र के लिए 0 × 0000-0xFFFF (0-65535) की सीमा होती है। लंबे समय तक टाइमर की सबसे अधिक कीमत कुछ अनुप्रयोगों और सर्किट में उपयोगी साबित हो सकती है। यह स्वचालित रूप से दो 16 बिट संख्या पर कार्य कर सकता है। 16-बिट माइक्रोकंट्रोलर के कुछ उदाहरण 16-बिट MCUs हैं, जिन्हें 8051XA, PIC2x, Intel 8096 और Motorola MC68HC12 तक विस्तारित किया गया है।

32-बिट माइक्रोकंट्रोलर 32-बिट इंस्ट्रक्शन का उपयोग Arihmetic और Logical संचालन के लिए किया जाता है। इनका उपयोग इम्प्लांटेबल मेडिकल डिवाइस, इंजन कंट्रोल सिस्टम, ऑफिस मशीन, उपकरण और अन्य प्रकार के एम्बेडेड सिस्टम सहित स्वचालित रूप से नियंत्रित उपकरणों में किया जाता है। कुछ उदाहरण Intel / Atmel 251, PIC3x हैं।

1.4.2 मेमोरी डिवाइस के आधार पर माइक्रोकंट्रोलर के प्रकार (Types of Microcontrollers Based on Memory Devices)

मेमोरी उपकरणों को दो वर्गों में विभाजित किया जाता है, ये हैं—

1. एम्बेडेड मेमोरी माइक्रोकंट्रोलर (Embedded memory micro-controller)
2. एक्सटर्नल मेमोरी माइक्रोकंट्रोलर (External memory micro-controller)

1. एम्बेडेड मेमोरी माइक्रोकंट्रोलर (Embedded Memory Micro-controller)—जब एक एम्बेडेड सिस्टम में एक माइक्रोकंट्रोलर यूनिट होती है, जिसमें चिप पर उपलब्ध सभी कार्यात्मक ब्लॉक माइक्रोकंट्रोलर यूनिट से कनेक्ट होते हैं। एक एम्बेडेड माइक्रोकंट्रोलर कहलाते हैं। उदाहरण के लिए, 8051 वाले प्रोग्राम और डाटा मेमोरी, I/O पोर्ट, सीरियल संचार, काउंटर और टाइमर और चिप पर इंटरप्ट एक एम्बेडेड माइक्रोकंट्रोलर है।

2. एक्सटर्नल मेमोरी माइक्रोकंट्रोलर (External Memory Micro-controller)—जब एक एम्बेडेड सिस्टम में एक माइक्रोकंट्रोलर यूनिट होती है, जिसमें चिप पर उपलब्ध सभी कार्यात्मक ब्लॉक नहीं होते हैं, बाह्य मेमोरी माइक्रोकंट्रोलर कहलाता है। उदाहरण के लिए, 8031 में कोई प्रोग्राम मेमोरी नहीं है, यह चिप एक बाह्य मेमोरी माइक्रोकंट्रोलर है।

1.4.3 इंस्ट्रक्शन सेट के आधार पर माइक्रोकंट्रोलर के प्रकार (Types of Microcontrollers Based on Instruction Set)

CISC—CISC एक कॉम्प्लेक्स इंस्ट्रक्शन सेट कंप्यूटर होता है, यह प्रोग्रामर को बहुत-से निर्देशों के स्थान पर एक निर्देश का प्रयोग करने देता है।

RISC—RISC Reduced Instruction Set Computer होता है, इस प्रकार के इंस्ट्रक्शन सेट उद्योग के मानकों के लिए माइक्रोप्रोसेसर के डिजाइन को कम करते हैं। यह प्रत्येक निर्देश को किसी भी रजिस्टर पर संचालित करने या किसी भी एड्रेसिंग मोड का उपयोग करने की अनुमति देता है और साथ ही साथ प्रोग्राम और डाटा का उपयोग करता है।

CISC :	Mov AX, 4	**RISC :**		Mov AX, 0
	Mov BX, 2			Mov BX, 4
	ADD BX, AX			Mov CX, 2
			Begin	ADD AX, BX
			Loop	Begin

CISC और RISC के उदाहरण—उपरोक्त उदाहरण से RISC सिस्टम प्रति निर्देश क्लॉक चक्र को कम करके निष्पादन समय कम करता है, और CISC सिस्टम प्रति प्रोग्राम निर्देशों की संख्या को कम करके निष्पादन समय को कम करता है। RISC ,CISC से बेहतर निष्पादन देता है।

1.4.4 मेमोरी आर्किटेकचर के आधार पर माइक्रोकंट्रोलर के प्रकार (Types of Micro-controllers Based on Memory Architecture)

माइक्रोकंट्रोलर की मेमोरी आर्किटेक्चर दो प्रकार की होती है, ये हैं—

- **हार्वर्ड मेमोरी आर्किटेकचर माइक्रोकंट्रोलर** (Harvard memory architecture micro-controller)
- **प्रिन्सटन मेमोरी आर्किटेकचर माइक्रोकंट्रोलर** (Princeton memory architecture micro-controller)

हार्वर्ड मेमोरी आर्किटेकचर माइक्रोकंट्रोलर (Harvard Memory Architecture Micro-controller)—वह बिंदु जब एक माइक्रोकंट्रोलर यूनिट में प्रोग्राम और डाटा मेमोरी के लिए एक असमान मेमोरी एड्रेस स्पेस होता है, तो माइक्रोकंट्रोलर के प्रोसेसर में हार्वर्ड मेमोरी आर्किटेक्चर होता है।

प्रिन्सटन मेमोरी आर्किटेकचर माइक्रोकंट्रोलर (Princeton Memory Architecture Micro-controller)—वह बिंदु जब प्रोग्राम मेमोरी और डाटा मेमोरी के लिए एक माइक्रोकंट्रोलर में एक सामान्य मेमोरी एड्रेस होता है, तो प्रोसेसर में माइक्रोकंट्रोलर प्रिंसटन मेमोरी आर्किटेक्चर होता है।

1.5 8051 माइक्रोकंट्रोलर (8051 Micro-controller)

तकनीकी रूप से इसे INTEL MCS-51 आर्किटेक्चर के रूप में जाना जाता है, 8051 माइक्रोकंट्रोलर श्रृंखला को INTEL द्वारा सन् 1980 में विकसित किया गया था और 80 के दशक में यह बहुत लोकप्रिय था (अभी भी लोकप्रिय हैं)। 8051 माइक्रोकंट्रोलर में श्रृंखला कम्युनिकेशन (series communication), टाइमर, इंटरप्ट इत्यादि जैसी कई विशेषताएँ हैं और इसलिए बहुत-से छात्र और लोग 8051 माइक्रोकंट्रोलर के साथ माइक्रोकंट्रोलर की अवधारणा पर अपना कार्य शुरू करते हैं (हालांकि यह प्रवृत्ति Arduino की शुरूआत के साथ बदली हुई लगती है)।

यद्यपि 8051 माइक्रोकंट्रोलर का प्रचलन कम है, परन्तु यह माइक्रोकंट्रोलर्स, एंबेडेड सिस्टम और प्रोग्रामिंग (सी और असेंबली दोनों) के साथ शुरू करने के लिए सबसे अच्छे प्लेटफार्मों में से एक है।

सेट करते है। जब Service Branch को Interrupt Service Routine (ISR) में भेजा जाता है, तो इंटरप्ट फ्लैग साफ़ हो जाते हैं। जब बाहरी शाखाएँ बाधित हो जाती हैं, और प्रोसेसर शाखाएँ रुकावट सेवा के रूट पर पहुँचती हैं, तो Interrupt एक नकारात्मक बढ़त प्रदान करता है, जबकि टाइमर और सीरियल पोर्ट में Interrupt होता है, उनमें से दो बाहरी Interrupt होते हैं, और दो टाइमर इंटरप्ट होते हैं और एक सीरियल पोर्ट इंटरप्ट टर्मिनल सामान्य रूप में होता है।

3. मेमोरी (Memory)

माइक्रोकंट्रोलर को एक प्रोग्राम की आवश्यकता होती है, जो निर्देशों का संग्रह होता है। यह कार्यक्रम विशिष्ट प्रोग्राम को करने के लिए माइक्रोकंट्रोलर को बताता है। इन प्रोग्राम्स के लिए एक मेमोरी की आवश्यकता होती है, जिस पर किसी विशेष कार्य के विशिष्ट प्रोग्राम को करने के लिए माइक्रोकंट्रोलर द्वारा इन्हें बचाया और पढ़ा जा सकता है। माइक्रोकंट्रोलर के प्रोग्राम को स्टोर करने के लिए उपयोग की जाने वाली मेमोरी को कोड मेमोरी या एप्लिकेशन की प्रोग्राम मेमोरी के रूप में जाना जाता है। इसे माइक्रोकंट्रोलर की ROM भी कहते हैं, इसके लिए अस्थायी रूप से माइक्रोकंट्रोलर में डाटा या ऑपरेंड को स्टोर करने के लिए मेमोरी की भी आवश्यकता होती है। 8051 के डाटा मेमोरी का उपयोग ऑपरेशन के लिए अस्थायी रूप से डाटा को स्टोर करने के लिए किया जाता है, जिसे RAM कहते हैं। 8051 माइक्रोकंट्रोलर में 4K मेमोरी या प्रोग्राम मेमोरी होती है, जिसमें 4KB ROM होती है और रैम की डाटा मेमोरी का 128 बाइट भी होती है।

4. BUS

मूल रूप से Bus तारों का एक समूह होता है, जो डेटा हस्तांतरण के लिए संचार चैनल या माध्यम के रूप में कार्य करता है। इन बसों में 8, 16 या अधिक माइक्रोकंट्रोलर के तार होते हैं। इस प्रकार, ये 8 बिट्स, 16 बिट्स को एक साथ ले जा सकते हैं। इसमें दो प्रकार की Bus होती है, जो निम्नलिखित होती हैं—

(i) एड्रेस बस (Address Bus)

(ii) डाटा बस (Data Bus)

(i) Address Bus—डाटा को ट्रांसफर करने के लिए माइक्रोकंट्रोलर 8051 में 16 बिट एड्रेस बस होती है। इसका उपयोग मेमोरी स्थानों को संबोधित करने और सीपीयू से मेमोरी को माइक्रोकंट्रोलर की मेमोरी को स्थानांतरित करने के लिए किया जाता है। इसके चार एड्रेसिंग मोड हैं जो इस प्रकार हैं।

(a) Immediate addressing modes.

(b) Bank address (or) Register addressing mode.

(c) Direct Addressing mode.

(d) Register indirect addressing mode.

(ii) Data Bus—माइक्रोकंट्रोलर 8051 में डाटा बस के 8 बिट्स होते हैं, जिनका उपयोग विशेष अनुप्रयोगों के डाटा को ले जाने के लिए किया जाता है।

5. ओस्सिलाटर (Oscillator)

हम जानते हैं, कि माइक्रोकंट्रोलर एक उपकरण है, इसलिए इसे माइक्रोकंट्रोलर अनुप्रयोगों के संचालन के लिए क्लॉक की पल्स की आवश्यकता होती है। इस उद्देश्य के लिए, माइक्रोकंट्रोलर 8051 में एक ऑन-चिप ओस्सिलेटर होता है, जो माइक्रोकंट्रोलर के सेंट्रल प्रोसेसिंग यूनिट के लिए क्लॉक के स्रोत के रूप में कार्य करता है। ओस्सिलेटर का उत्पादित पल्स स्थिर होता हैं। इसलिए, यह 8051 माइक्रोकंट्रोलर के सभी भागों के सिंक्रनाइज़ कार्य को सक्षम करता है।

6. Input/Output Port

सामान्यत: माइक्रोकंट्रोलर में मशीनों के संचालन को नियंत्रित करने के लिए एम्बेडेड सिस्टम में माइक्रोकंट्रोलर का उपयोग किया जाता है। इसलिए, इसे अन्य मशीनों, उपकरणों या बाह्य उपकरणों से जोड़ने के लिए हमें माइक्रोकंट्रोलर

इंटरफ़ेस में I / O इंटरफेसिंग पोर्ट की आवश्यकता होती है। इस प्रयोजन के लिए माइक्रोकंट्रोलर 8051 में 4 इनपुट, आउटपुट पोर्ट होते हैं जो इसे अन्य बाह्य उपकरणों से जोड़ने के लिए हैं

7. Timers/Counters

8051 माइक्रोकंट्रोलर में दो 16 बिट के टाइमर और काउंटर होते हैं। इन काउंटरों को पुन: 8 बिट रजिस्टर में विभाजित किया गया है। पल्स की चौड़ाई निर्धारित करने के लिए अंतराल के मापन के लिए टाइमर का उपयोग किया जाता है।

8051 माइक्रोकंटोलर आर्किटेक्चर की विशेषताएँ—

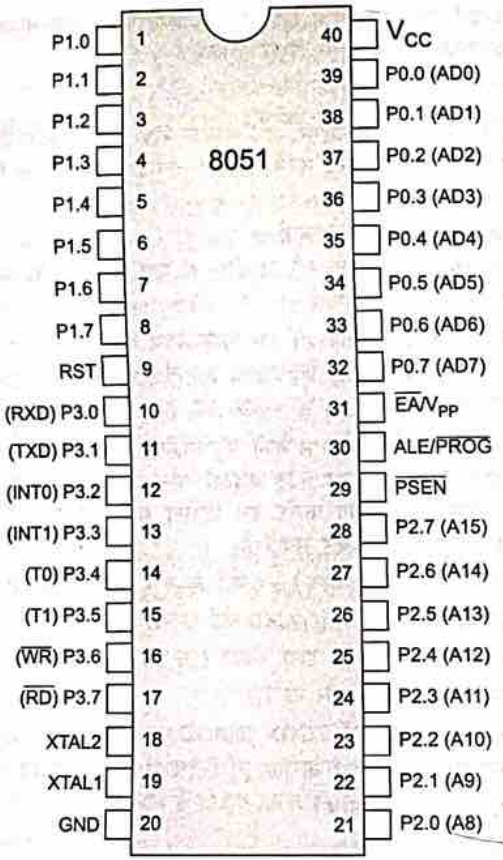
हमने उपरोक्त खंड में 8051 माइक्रोकंट्रोलर की आंतरिक संरचना को देखा है। अब, हम 8051 माइक्रोकंट्रोलर आर्किटेक्चर की विशेषताओं को देखेंगे जो इस प्रकार है—

- इसमें 8-दो रजिस्टर A (Accumulator) और B के साथ बिट CPU होते हैं।
- **8K बाइट्स का आंतरिक रोम**—यह एक फ्लैश मेमोरी है, जो सिस्टम प्रोग्रामिंग में समर्थन करता है।
- **256 बाइट्स की आंतरिक रैम**—रैम के पहले 128 बाइट्स यानी 00H से 7FH को फिर से प्रत्येक बैंक में 8 रजिस्टर (R0-R7) के साथ 4 बैंकों में विभाजित किया जाता है, 16 बिट एड्रेस रजिस्टर और 80% उद्देश्य रजिस्टर। रैम के उच्च 128 बाइट्स यानी 80H से FFH में SFR या विशेष फ़ंक्शन रजिस्टर होते हैं। SFR का उपयोग करके हम विभिन्न बाह्य उपकरणों; जैसे—टाइमर, सीरियल पोर्ट, सभी I/O पोर्ट आदि को नियंत्रित कर सकते हैं।
- **32 I/O पिन (इनपुट / आउटपुट पिन)**—4 पोर्ट के रूप में व्यवस्थित: P0, P1, P2 और P3।
- 8-बिट स्टैक पॉइंटर (SP) और प्रोसेसर स्टेटस वर्ड (PSW)।
- 16-बिट प्रोग्राम काउंटर (पीसी) और डाटा पॉइंटर (DPTR)।
- दो 16-बिट टाइमर / काउंटर-T_0 और टी T_1।
- नियंत्रण रजिस्टर—SCON, PCON, TCON, TMOD, IP और IE।
- सीरियल डाटा ट्रांसमीटर और रिसीवर के लिए पूर्ण-द्वैध संचालन-SBUF।
- व्यवधान: दो बाहरी और तीन आंतरिक।
- ऑस्सिलेटर और क्लॉक सर्किट।

1.7 8051 माइक्रोकंट्रोलर का पिन डायग्राम (Pin Diagram of 8051 Micro-controller)

8051 माइक्रोकंट्रोलर (89C51, 8751, DS89C4xO, 89C52) क्वाड-फ्लैट पैकेज (quad-flat package), लीडलेस चिप कैरियर (leadless chip carrier) और डुअल-इन-लाइन पैकेज (dual-in-line package) जैसे विभिन्न पैकेजों में आते हैं। इन सभी पैकेजों में 40 पिन होती हैं, जो I/O, एड्रेस, RD, WR, डाटा और इंटरप्ट जैसे कई कार्यों के लिए समर्पित होती हैं। लेकिन, कुछ कम्पनियाँ I/O पोर्ट की संख्या को कम करके अनुप्रयोगों की मांग के लिए माइक्रोकंट्रोलर्स का 20-पिन का संस्करण पेश करती हैं। फिर भी, अधिकांश डेवलपर्स 40-पिन चिप का उपयोग करते हैं।

8051 माइक्रोकंट्रोलर के पिन आरेख में 40 पिन होती हैं जैसा कि चित्र में दिखाया गया है। 32 पिन को चार पोर्ट्स; जैसे—P0, P1, P2 और P3 में सेट किया जाता है। जहाँ, प्रत्येक पोर्ट में 8 पिन होती हैं। माइक्रोकंट्रोलर 8051 का पिन आरेख और स्पष्टीकरण को चित्र 1.5 में दर्शाया गया है।



चित्र 1.5 : माइक्रोकंट्रोलर का पिन आरेख

- **पोर्ट 1 (पिन 1 से पिन 8)**—पोर्ट 1 में Pin1.0 से Pin1.7 होती हैं और इन पिनों को इनपुट या आउटपुट पिन के रूप में कॉन्फ़िगर किया जा सकता है।
- **पिन 9** (RST)—इस पिन को एक पॉजिटिव पल्स देकर 8051 माइक्रोकंट्रोलर को रीसेट करने के लिए रीसेट पिन का उपयोग किया जाता है।
- **पोर्ट 3 (पिन 10 से 17)**—पोर्ट 3 पिन पोर्ट 1 पिन के समान है और इसे यूनिवर्सल इनपुट या आउटपुट पिन के रूप में उपयोग किया जा सकता है। ये पिन दोहरे-कार्य पिन और प्रत्येक पिन के कार्य के रूप में दिए गए हैं।
- **पिन 10 (RXD)**—RXD पिन एक सीरियल एसिंक्रोनस कम्युनिकेशन इनपुट या सीरियल सिंक्रोनस कम्युनिकेशन आउटपुट है।
- **पिन 11 (TXD)**—सीरियल असिंक्रोनस कम्युनिकेशन आउटपुट या सीरियल सिंक्रोनस कम्युनिकेशन क्लॉक आउटपुट।
- **पिन 12** (INT0)—इंटरप्ट 0 का इनपुट

- **पिन 13** (INT1)—इंटरप्ट 1 का इनपुट
- **पिन 14** (T0)—काउंटर 0 घड़ी का इनपुट
- **पिन 15** (T1)—काउंटर 1 घड़ी का इनपुट
- **पिन 16** (WR)—बाह्य रैम पर कंटेंट लिखने के लिए सिग्नल लिखना।
- **पिन 17** (RD)—बाह्य रैम की कंटेंट को पढ़ने के लिए सिग्नल पढ़ना।
- **पिन 18 और 19 (XTAL2, XTAL1)**—X2 और X1 पिन ऑसिलेटर के लिए इनपुट आउटपुट पिन हैं। इन पिनों का उपयोग माइक्रोकंट्रोलर को आंतरिक ऑसिलेटर से जोड़ने के लिए किया जाता है।
- **पिन 20** (GND)—पिन 20 एक ग्राउंड पिन है।
- **पोर्ट 2 (पिन 21 से पिन 28)**—पोर्ट 2 में पिन 21 से पिन 28 शामिल हैं, जिन्हें इनपुट/आउटपुट पिन के रूप में कॉन्फ़िगर किया जा सकता है। लेकिन, यह तभी संभव है, जब हम किसी बाह्य मेमोरी का उपयोग नहीं करेंगे। यदि हम बाह्य मेमोरी का उपयोग करते हैं, तो ये पिन उच्च क्रम एड्रेस बस (A8 से A15) के रूप में कार्य करेंगे।
- **पिन 29** (PSEN)—इस पिन का उपयोग बाह्य प्रोग्राम मेमोरी को सक्षम बनाने के लिए किया जाता है। यदि हम प्रोग्राम को संग्रहीत करने के लिए एक बाह्य ROM का उपयोग करते हैं, तो उस पर लॉजिक 0 दिखाई देता है, जो मेमोरी से डाटा पढ़ने के लिए माइक्रो नियंत्रक को इंगित करता है।
- **पिन 30** (ALE)—Address Latch Enable pin पिन एक सक्रिय उच्च-आउटपुट संकेत होता है। यदि हम बहुत-सी मेमोरी चिप का उपयोग करते हैं, तो इस पिन का उपयोग उनके बीच अंतर करने के लिए किया जाता है। यह पिन EPROM की प्रोग्रामिंग के दौरान प्रोग्राम पल्स इनपुट भी देता है।
- **पिन 31 (EA)**—यदि हमें बहुत-सी मेमोरी का उपयोग करना है, तो इस पिन के लिए लॉजिक 1 का एप्लीकेशन दोनों मेमोरी से डाटा को पढ़ने के लिए माइक्रोकंट्रोलर को निर्देश देता है: पहले आंतरिक और फिर बाहरी।
- **पोर्ट 0 (पिन 32 से 39)**—पोर्ट 2 और 3 पिन के समान, इस पिन का उपयोग इनपुट/आउटपुट पिन के रूप में किया जा सकता है। जब हम किसी बाह्य मेमोरी का उपयोग नहीं करते हैं। जब ALE या Pin 30, 1 पर होता है, तो इस पोर्ट का उपयोग डाटा बस के रूप में किया जाता है। जब ALE पिन 0 पर होता है, तो इस पोर्ट का उपयोग एक लोअर ऑर्डर एड्रेस बस (A0 से A7) के रूप में किया जाता है
- **पिन 40** (VCC)—इस VCC पिन का उपयोग बिजली की आपूर्ति के लिए किया जाता है।

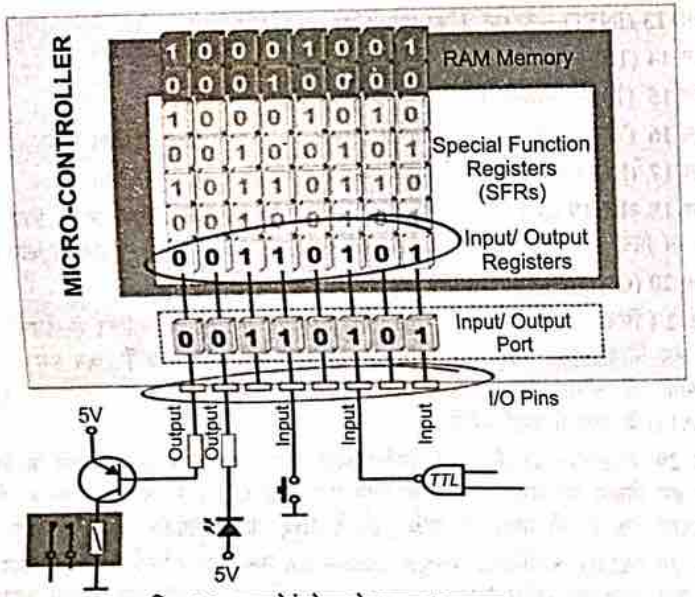
1.8 माइक्रोकंट्रोलर 8051 के इनपुट/आउटपुट पोर्ट (Micro-controllers 8051 Input Output Ports)

8051 माइक्रोकंट्रोलर्स में 8-बिट के 4 I/O पोर्ट होते हैं। इसलिए, कुल 32 इनपुट/आउटपुट पिन पेरिफेरेल उपकरणों के साथ माइक्रोकंट्रोलर को जोड़ने की अनुमति देते हैं।

पिन कॉन्फ़िगरेशन, यानी पिन को लॉजिक स्थिति के अनुसार आउटपुट के लिए 1 और आउटपुट के लिए 0 के रूप में कॉन्फ़िगर किया जा सकता है।

इनपुट/आउटपुट (I/O) पिन

माइक्रोकंट्रोलर में सभी सर्किट P0 पोर्ट को छोड़कर इसके किसी एक पिन से जुड़े होने चाहिए क्योंकि इसमें पुल-अप रेसिस्टर्स बिल्ट-इन नहीं होते हैं।



चित्र 1.6 : माइक्रोकंट्रोलर के इनपुट/आउटपुट

इनपुट पिन

लॉजिक 1 P रजिस्टर के एक बिट पर लागू होता है। आउटपुट FE ट्रांजिस्टर को बंद कर दिया जाता है और दूसरा पिन उच्च प्रतिरोध के पुल-अप प्रतिरोध पर विद्युत की आपूर्ति वोल्टेज से जुड़ा रहता है।

पोर्ट 0-P0 (शून्य) पोर्ट के दो कार्यों हैं—

- जब बाह्य मेमोरी का उपयोग किया जाता है, तो उस पर निचला एड्रेस बाइट (एड्रेस A0A7) लगाया जाता है, अन्यथा इस पोर्ट के सभी बिट्स इनपुट/आउटपुट के रूप में कॉन्फ़िगर किए जाते हैं।
- जब P0 पोर्ट को आउटपुट का आकार दिया जाता है, तो अन्य पोर्ट्स जिसमें अंतर्निर्मित पुल-अप होता है, जो इसके 5V की आपूर्ति से जुड़ा होता है, इस पोर्ट की पिनों में यह रेसिस्टर होता है।

इनपुट कॉन्फ़िगरेशन

यदि इस पोर्ट की कोई भी पिन इनपुट के रूप में कॉन्फ़िगर की गई है, तो यह कार्य करता है, जैसे कि यह 'फ्लोट' करता है, यानी इनपुट में असीमित इनपुट प्रतिरोध और इन-निर्धारित क्षमता होती है।

आउटपुट कॉन्फ़िगरेशन

जब पिन आउटपुट के रूप में कॉन्फ़िगर किया जाता है, तो यह 'ओपन ड्रेन' के रूप में कार्य करता है। लॉजिक 0 को पोर्ट बिट पर लागू करने से, उपयुक्त पिन ग्राउंड (0V) से जुड़ा होगा, और लॉजिक 1 को लागू करने पर, बाह्य आउटपुट 'चालू' रहेगा। इस आउटपुट पिन पर लॉजिक 1 (5V) लागू करने के लिए, बाह्य पुलअप अवरोधक का निर्माण करना आवश्यक होता है।

पोर्ट 1—P1 एक I/O पोर्ट है, क्योंकि इसमें P0 के समान कोई वैकल्पिक फंक्शन नहीं होता है, लेकिन इस पोर्ट को केवल सामान्य I/O के रूप में कॉन्फ़िगर किया जा सकता है। इसमें एक अंतर्निहित पुल-अप रेसिस्टर होता है और पूरी तरह से TTL सर्किट के साथ संगत होता है।

पोर्ट 2— P2, P0 के समान है जब बाहरी मेमोरी का उपयोग किया जाता है। इस पोर्ट के पिन बाह्य मेमोरी चिप के लिए निर्धारित एड्रेस पर अधिकार कर लेते हैं। इस पोर्ट का उपयोग A8-A15 के एड्रेस के साथ उच्च एड्रेस बाइट के लिए किया जा सकता है। जब कोई मेमोरी नहीं जोड़ी जाती है तो इस पोर्ट को पोर्ट 1 के समान सामान्य इनपुट / आउटपुट पोर्ट के रूप में उपयोग किया जा सकता है।

पोर्ट 3—इस पोर्ट के, फ़ंक्शन अन्य पोर्ट के समान हैं सिवाय इसके कि लॉजिक 1 को P3 रजिस्टर के उपयुक्त बिट पर लागू किया जाना चाहिए।

1.9 पिन करंट सीमाएँ (Pins Current Limitations)

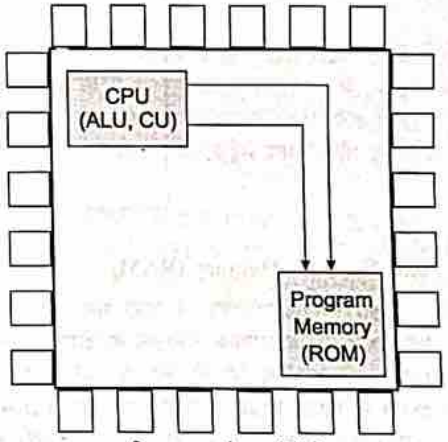
- जब पिंन को आउटपुट के रूप में कॉन्फ़िगर किया जाता है (यानी लॉजिक 0), तो सिंगल पोर्ट पिन को 10mA का करंट मिल सकता है।
- जब इन पिनों को इनपुट (यानी लॉजिक 1) के रूप में कॉन्फ़िगर किया जाता है, तो अंतर्निहित पुल-अप प्रतिरोध बहुत कम करंट प्रदान करते हैं, लेकिन LS श्रृंखला के 4 टीटीएल इनपुट तक सक्रिय कर सकते हैं।
- यदि पोर्ट के सभी 8 बिट सक्रिय होते हैं, तो कुल करंट 15 mA (पोर्ट P0: 26 mA) तक सीमित होना चाहिए।
- यदि पोर्ट के सभी (32 बिट) सक्रिय हैं, तो कुल अधिकतम करंट 71 mA तक सीमित होना चाहिए।

1.10 8051 माइक्रोकंट्रोलर का मेमोरी आर्गेनाइजेशन (8051 Micro-controller Memory Organization)

8051 माइक्रोकंट्रोलर मेमोरी को प्रोग्राम मेमोरी (ROM) और डाटा मेमोरी (RAM) में अलग किया जाता है। 8051 माइक्रोकंट्रोलर के प्रोग्राम मेमोरी को प्रोग्राम को निष्पादित करने के लिए उपयोग किया जाता है यानी इंस्ट्रक्शन। दूसरी ओर डाटा मेमोरी, अस्थायी वेरिएबल डाटा और मध्यवर्ती परिणामों को संग्रहीत करने के लिए उपयोग की जाती है। 8051 माइक्रोकंट्रोलर में आंतरिक ROM और आंतरिक RAM दोनों होते हैं। यदि आंतरिक मेमोरी अपर्याप्त है, तो आप उपयुक्त सर्किट का उपयोग करके बाहरी मेमोरी जोड़ सकते हैं।

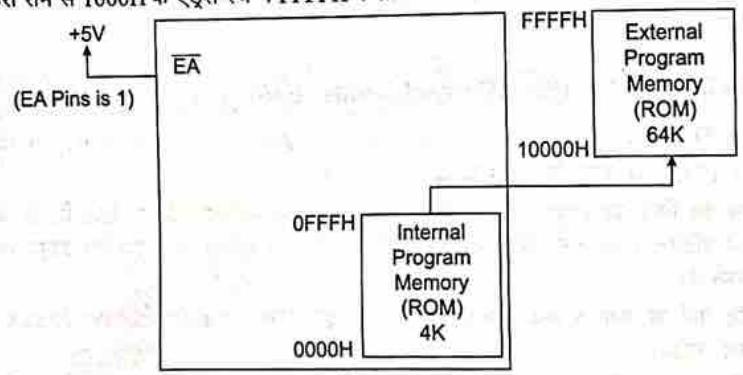
प्रोग्राम मेमोरी Program Memory (ROM)

8051 माइक्रोकंट्रोलर में निष्पादित होने वाले कोड या निर्देशों को प्रोग्राम मेमोरी में संग्रहीत किया जाता है, जिसे माइक्रोकंट्रोलर की RAM भी कहा जाता है। इंटेल द्वारा मूल 8051 माइक्रोकंट्रोलर में आंतरिक ROM में 4KB मेमोरी होती है। श्रृंखला 8051 के कुछ वेरिएंट; जैसे—8031 और 8032 में कोई आंतरिक रोम (प्रोग्राम मेमोरी) नहीं होता है और इसमें लोड किए गए निर्देशों के साथ बाहरी प्रोग्राम मेमोरी के साथ हस्तक्षेप किया जाना चाहिए। लगभग सभी आधुनिक 8051 माइक्रोकंट्रोलर्स, जैसे कि 8052 श्रृंखला में 8KB की इंटरनल प्रोग्राम मेमोरी (ROM) होती है, जो फ्लैश मेमोरी (ROM) के रूप में होती है और मेमोरी को रीप्रोग्राम करने का विकल्प प्रदान करती है।



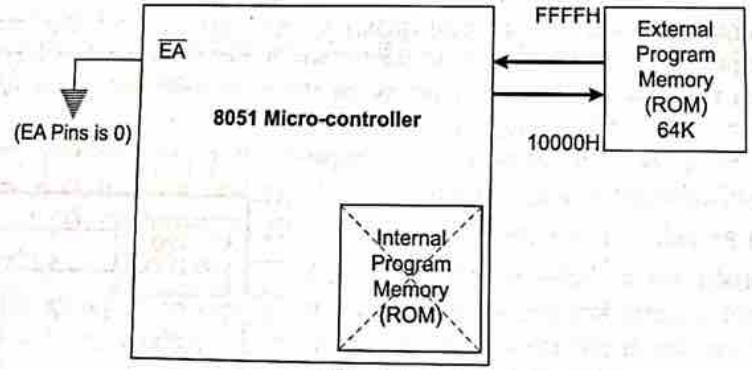
चित्र 1.7 : प्रोग्राम मेमोरी

आंतरिक ROM के 4KB में, एड्रेस प्लेस 0000H से 0FFFH होते है। यदि एड्रेस प्लेस यानी प्रोग्राम एड्रेस इस मान से अधिक है, तो CPU स्वचालित रूप से बाहरी प्रोग्राम मेमोरी से कोड फेच करेगा। इसके लिए, बाह्य एक्सेस पिन (EA पिन) को उच्च पुल करना चाहिए अर्थात जब EA पिन हाई होता है, तो सीपीयू पहले 0000H से 0FFFH के एड्रेस रेंज में आंतरिक प्रोग्राम मेमोरी से निर्देश प्राप्त करता है और यदि मेमोरी एड्रेस की सीमा से अधिक है, फिर निर्देशों को बाहरी रॉम से 1000H के एड्रेस रेंज में FFFFH में लाया जाता है।



चित्र 1.8

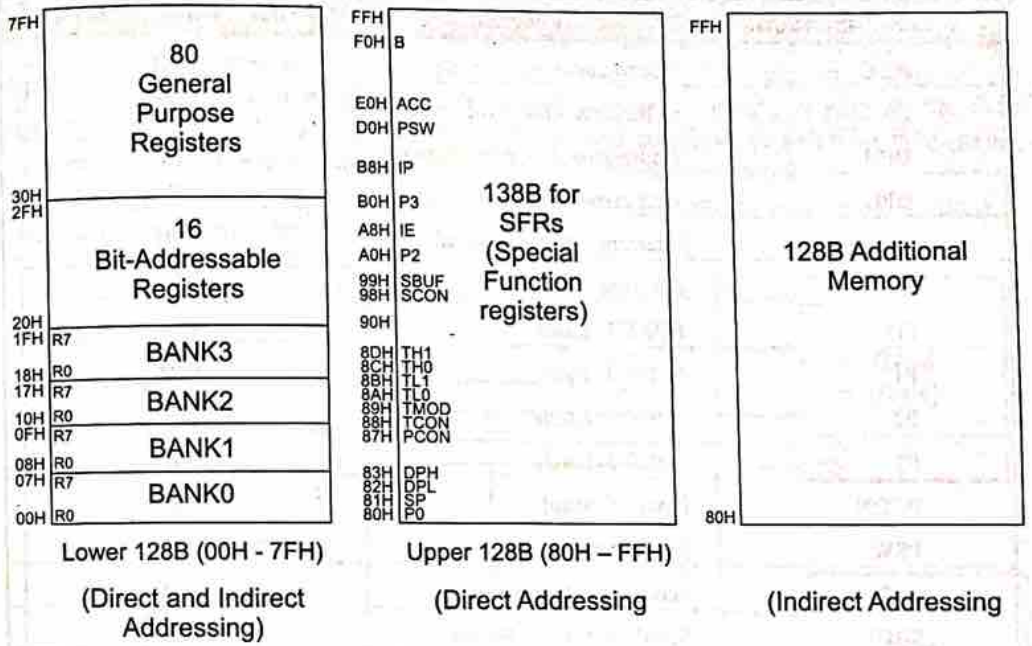
निर्देशों को लाने का एक और तरीका है: आंतरिक रोम को अनदेखा करना और केवल बाहरी प्रोग्राम मेमोरी (बाह्य रोम) से सभी निर्देशों को प्राप्त करना। इस परिदृश्य के लिए, EA पिन GND से जुड़ा होना चाहिए। इस स्थिति में बाहरी ROM का मेमोरी एड्रेस 0000H से FFFFH होगा।



चित्र 1.9

डाटा रैम Data Memory (RAM)

8051 माइक्रोकंट्रोलर के डाटा मेमोरी या रैम में अस्थायी डाटा और मध्यवर्ती परिणाम संग्रहीत होते हैं जो कि माइक्रोकंट्रोलर के सामान्य संचालन के दौरान उत्पन्न और उपयोग किए जाते हैं। मूल इंटेल के 8051 माइक्रोकंट्रोलर में 128 B की आंतरिक रैम थी। लेकिन 8051 माइक्रोकंट्रोलर के लगभग सभी आधुनिक वेरिएंट में 256B RAM है। इस 256B में, पहला 128B यानी मेमोरी एड्रेस 00H से 7FH तक वर्किंग रजिस्टर (रजिस्टर बैंक के रूप में संगठित), बिट - एड्रेसेबल एरिया और जनरल पर्पस रैम (जिसे स्क्रैचपैड क्षेत्र के रूप में भी जाना जाता है) में विभाजित किया गया है। रैम के पहले 128 बी (00H से 7FH) में, पहले 32 B यानी एड्रेस 00H से 7FH तक मेमोरी में 32 वर्किंग रजिस्टर होते हैं, जो प्रत्येक बैंक में 8 रजिस्टरों वाले चार बैंकों के रूप में व्यवस्थित होते हैं।



चित्र 1.10 : डाटा रैम

इसमें 4 बैंकों का नाम Bank0, Bank1, Bank2 और Bank3 हैं। प्रत्येक बैंक में R0 - R7 के नाम से 8 रजिस्टर होते हैं। प्रत्येक रजिस्टर को दो तरीकों से संबोधित किया जा सकता है—या तो नाम से या पते से। रजिस्टर को नाम से संबोधित करने के लिए, पहले संबंधित बैंक को चुनना होगा। बैंक का चयन करने के लिए, हमें RS0 और RS1 बिट्स का उपयोग करना होगा। उदाहरण के लिए अपने एड्रेस यानी 12H का उपयोग करके रजिस्टर को संबोधित करते समय, संबंधित बैंक चयनित हो सकता है या नहीं। (12H Bank2 में R2 से मेल खाती है)।

RAM का अगला 16B यानी 20H से 2FH बिट - एड्रेसेबल मेमोरी लोकेशन हैं। इसमें कुल 128 बिट्स हैं, जिन्हें व्यक्तिगत रूप से 00H से 7FH का उपयोग करके परिभाषित किया जा सकता है या पूरे बाइट को 20H से 2FH के रूप में परिभाषित किया जा सकता है। उदाहरण के लिए 32H आंतरिक रैम स्थान 26H का बिट 2 है। आंतरिक RAM का अंतिम 80B यानि 30H से 7FH तक का एड्रेस , सामान्य उद्देश्य RAM क्षेत्र है, जो कि एड्रेस इनेबल हैं। इन कम 128B RAM को प्रत्यक्ष या अप्रत्यक्ष रूप से संबोधित किया जा सकता है।

RAM का ऊपरी 128B यानी मेमोरी एड्रेस 80H से FFH तक स्पेशल फंक्शन रजिस्टर (SFR) के लिए आवंटित किया जाता है। SFR 8051 माइक्रोकंट्रोलर के विशिष्ट कार्यों को नियंत्रित करता है। SFR के कुछ I/O पोर्ट रजिस्टर (P0, P1, P2 और P3), PSW (प्रोग्राम स्टेटस वर्ड), A (Accumulator), IE (इंटरप्ट इनेबल), PCON (पॉवर कंट्रोल), आदि होते हैं।

Name of the Register	Function	Internal RAM Address (HEX)
ACC	Accumulator	EOH
B	B Register (for Arithmetic)	FOH
DPH	Addressing External Memory	83H
DPL	Addressing External Memory	82H
IE	Interrupt Enable Control	ASH
IP	Interrupt Priority	B8H
PO	PORT 0 Latch	80H
P1	PORT 1 Latch	90H
P2	PORT 2 Latch	AOH
P3	PORT 3 Latch	BOH
PCON	Power Control	87H
PSW	Program Status Word	DOH
SCON	Serial Port Control	98H
SBUF	Serial Port Data Buffer	99H
SP	Stack Pointer ILlc	81H
TMOD	Timer/Counter Mode Control	89H
ICON	Timer/Counter Control	88H
TLO	Timer 0 LOW Byte	8AH
THO	Timer 0 HIGH Byte	8CH
TL1	Timer 1 LOW Byte	8BH
TH1	Timer 1 HIGH Byte	8DH

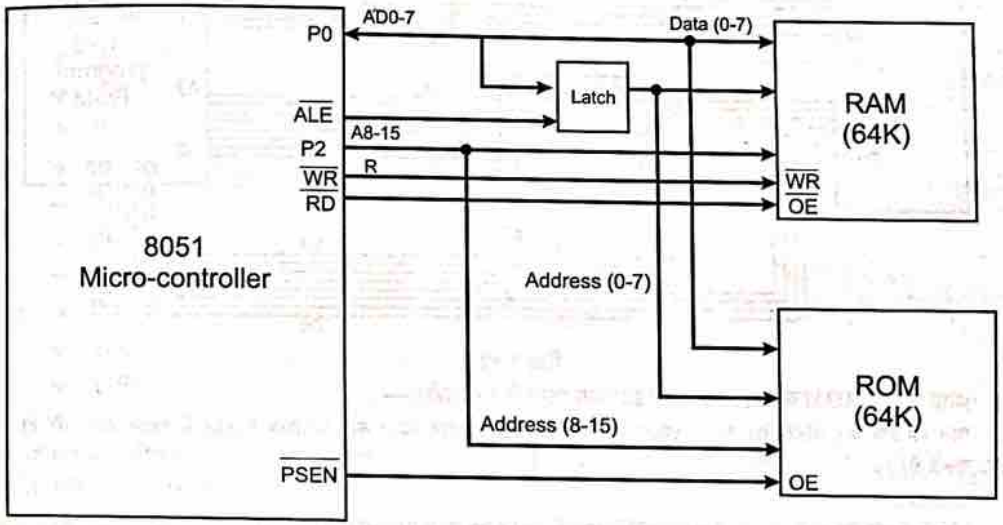
SRFs मेमोरी एड्रेस केवल Direct addressable होते हैं। भले ही 80H और FFH के बीच के कुछ एड्रेस किसी SFR को असाइन नहीं किए गए हैं, लेकिन उन्हें अतिरिक्त RAM क्षेत्र के रूप में उपयोग नहीं किया जा सकता है। कुछ माइक्रोकंट्रोलर्स में अतिरिक्त 128B RAM होता है, जो मेमोरी एड्रेस को SFRs यानी 80H से FFH के साथ साझा करता है। लेकिन यह अतिरिक्त RAM ब्लॉक केवल अप्रत्यक्ष रूप से संबोधित करके पहुँचा जाता है।

1.11 8051 माइक्रोकंट्रोलर के साथ बाह्य मेमोरी को इंटरफैस करना (Interfacing External Memory with 8051 Microcontroller)

हमने देखा है कि एक विशिष्ट 8051 माइक्रोकंट्रोलर में 4KB की RAM और 128B RAM होती है (सबसे आधुनिक 8051 माइक्रोकंट्रोलर वेरिएंट में 8K ROM और 256B RAM है)। 8051 माइक्रोकंट्रोलर आधारित प्रणाली का डिजाइनर 8051 माइक्रोकंट्रोलर में उपस्थित आंतरिक RAM और ROM तक सीमित नहीं हो सकता है। इसमें

External RAM और ROM यानी डाटा मेमोरी और प्रोग्राम दोनों को जोड़ने का प्रावधान होता है। बाह्य प्रोग्राम मेमोरी या ROM को इंटरफ़ेयर करने का कारण यह है, कि उच्च स्तरीय भाषाओं में लिखे गए जटिल प्रोग्राम अक्सर बड़े होते हैं और अधिक मेमोरी पर कब्जा कर लेते हैं। एक और महत्वपूर्ण कारण यह है, कि 8031 या 8032 जैसे चिप, जिनमें कोई Internal ROM नहीं है, External ROM के साथ हस्तक्षेप करना होगा। अधिकतम 64B प्रोग्राम मेमोरी (ROM) और डाटा मेमोरी (RAM) प्रत्येक को 8051 माइक्रोकंट्रोलर के साथ इंटरफेस किया जा सकता है।

दिया गया 8051 माइक्रोकंट्रोलर के साथ External RAM के 64KB और External ROM के 64KB को रोकने के ब्लॉक आरेख को दर्शाता है।



चित्र 1.11

Step 1 : Connect EA pin to ground

Step 2 : Connect the PSEN to the CE and OE.

Step 3 : Then, Port 2 (P2.0 – P2.7) to A8 – A12 pins of ext. ROM.

Step 4 : Connect ALE to G of 74LS373 latch to enable it.

Step 5 : Next, connect the OC of 74LS373 to GND.

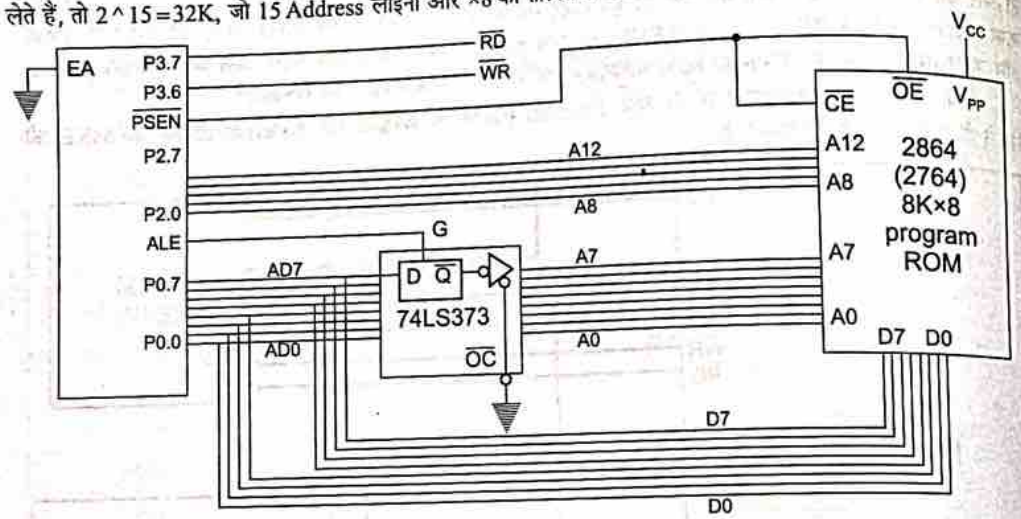
Step 6 : Connect Port 0 (P0.0 – P0.7), which consists of both address and data multiplexed into Port 0 to 1D – 8D pins of 74LS373 latch to demultiplex it and 1Q – 8Q of the latch to A0 – A7 of ext. ROM.

Step 7 : Connect Port 0 (P0.0 – P0.7) to D0 – D7 of the ext. ROM.

Step 8 : VPP of ext. ROM to VCC.

यहां, 8Kx8 का अर्थ है कि प्रोग्राम ROM एक ऐसी संरचना में आयोजित किया गया है, जिसमें 8K शब्द का स्थान है और x8 का अर्थ है, कि प्रत्येक शब्द में 8 बिट्स है। इसका अर्थ है कि 32K×8 एक ROM होगा, जिसमें 32K शब्द की जगह होगी, प्रति शब्द 8-बिट्स। अन्य शब्दों में, इसका अर्थ है कि 32,000 स्थान हैं जो 8-बिट चौड़े हैं। प्रोग्राम और डाटा मेमोरी 1K×8, 2K×8, 4K×8, 8K×8, 16K×8, 32K×8 और 64K×8 आकार की हो सकती है।

इसके अतिरिक्त, छोटे आकार के कई चिप एक साथ बड़े आकार की एक चिप बनाने के लिए कैस्केड करते हैं, यानी हम दो 16K×8 Data RAM Chip को एक 32K×8 Data RAM बनाने के लिए जोड़ सकते हैं। यदि हम 32K×8 लेते हैं, तो 2^15=32K, जो 15 Address लाइनों और ×8 का तात्पर्य 8 डाटा लाइनों से है।



चित्र 1.12

बाह्य प्रोग्राम ROM को 8051 के साथ इंटरफ़ेस करने के लिए कोड—

मान लो कि हम बाहरी ROM में स्थानों से 4000H तक संग्रहीत डाटा को आंतरिक RAM में स्थान 40H पर ले जाना चाहते हैं।

```
ORG 0000H
MOV DPTR, #4000H    ; Load DPTR with the location where data is stored
MOV R0, #40H        ; Load R0 with the int RAM loc where you want to save the data
rep: MOV A, #00H    ; Clear accumulator
MOVC A, @A+DPTR     ; Syntax to mode data from ext. ROM to accumulator
MOV @R0, A          ; Copy the value of accumulator in location pointed by R0
INC R0              ; Inc R0 to point to next int RAM location
INC DPTR            ; Inc DPTR to point to next ext. ROM location
CJNE A, #00H, rep   ; Repeat this process until 0 is received from the DPTR
stay:SJMP stay      ; Stay here
END                 ; Let's say that the data present at 4000H is
ORG 4000H
DB 1H, 2H, 0AH, 0F2H, 30H, 5CH, 2AH, 01H, 00H, FFH, 0
                    ; Here 0 is the stop bit that we're assuming
```

1.12 स्पेशल फंक्शन रजिस्टर्स (Special Function Registers (SFRs))

8051 माइक्रोकंट्रोलर स्पेशल फंक्शन रजिस्टर एक नियंत्रण तालिका के रूप में कार्य करता है, जो 8051

माइक्रोकंट्रोलर के संचालन की निगरानी और नियंत्रण करता है। यदि आप Internal RAM संरचना का निरीक्षण करते हैं, तो 80H से FFH तक का एड्रेस स्पेस SFR को आवंटित किया जाता है। इन 128 मेमोरी स्थानों में से (80H से FFH), केवल 21 स्थान हैं, जो वास्तव में SFRs को सौंपे गए हैं। प्रत्येक SFR में एक बाइट एड्रेस होता है और एक यूनिक नाम भी होता है जो इसके उद्देश्य को निर्दिष्ट करता है। चूंकि SFR Internal RAM की संरचना का एक भाग हैं, इसलिए आप एईं का उपयोग कर सकते हैं, जैसे कि आप Internal RAM का उपयोग करते हैं। मुख्य अंतर एड्रेस स्पेस (Address space) है: पहला 128 बाइट्स (00H से 7FH) नियमित Internal RAM के लिए है और अगला 128 बाइट्स (80H से FFH) SFRs के लिए है।

8051 माइक्रोकंट्रोलर स्पेशल फंक्शन रजिस्टर्स की सूची

- A or ACC
- DPL
- IE
- P0
- P2
- PCON
- SCON
- SP
- TCON
- TH0
- TH1
- B
- DPH
- IP
- P1
- P3
- PSW
- SBUF
- TMOD
- TL0
- TL1

8051 माइक्रोकंट्रोलर SFR की श्रेणियाँ—सभी 21 8051 माइक्रोकंट्रोलर SFRs उनके कार्यों और Internal RAM एड्रेस के साथ निम्नलिखित तालिका में दिए गए हैं।

Name of the Register	Function	Internal Ram Address (HEX)
ACC	Accumulator	EOH
B	B Register (for Arithmetic)	FOH
DPH	Addressing External Memory	83H
DPL	Addressing External Memory	82H
IE	Interrupt Enable Control	ASH
IP	Interrupt Priority	B8H
PO	PORT 0 Latch	80H
P1	PORT 1 Latch	90H
P2	PORT 2 Latch	AOH
P3	PORT 3 Latch	BOH
PCON	Power Control	87H
PSW	Program Status Word	DOH

SCON	Serial Port Control	98H
SBUF	Serial Port Data Buffer	99H
SP	Stack Pointer	81H
TMOD	Timer / Counter Mode Control	89H
ICON	Timer / Counter Control	88H
TLO	Timer 0 LOW Byte	8AH
THO	Timer 0 HIGH Byte	8CH
TL1	Timer 1 LOW Byte	8BH
TH1	Timer 1 HIGH Byte	8DH

इन 21 SFRs को वर्गीकृत करने के कई तरीके हैं, लेकिन इनमे निम्नलिखित तरीके जो उपयुक्त होते है। 8051 माइक्रोकंट्रोलर के 21 SFRs को सात समूहों में वर्गीकृत किया गया है। ये हैं—

Math or CPU Registers: A and B

Status Register: PSW (Program Status Word)

Pointer Registers: DPTR (Data Pointer – DPL, DPH) and SP (Stack Pointer)

I/O Port Latches: P0 (Port 0), P1 (Port 1), P2 (Port 2) and P3 (Port 3)

Peripheral Control Registers: PCON, SCON, TCON, TMOD, IE and IP

Peripheral Data Registers: TL0, TH0, TL1, TH1 and SBUF

CPU या मैथ रजिस्टर्स (CPU or Math Registers)

1. A or Accumulator (ACC)—Accumulator या Register A सबसे महत्वपूर्ण और सबसे अधिक उपयोग किया जाने वाला 8051 माइक्रोकंट्रोलर SFR है। रजिस्टर A SFR मेमोरी स्पेस में एड्रेस E0H पर स्थित होता है। Accumulator का उपयोग लगभग सभी ALU परिचालनों के डाटा को रखने के लिए किया जाता है।

Accumulator द्वारा उपयोग किए जाने वाले कुछ ऑपरेशन निम्नलिखित हैं—

- अंकगणित संचालन; जैसे—जोड़, घटा, गुणा आदि।
- लॉजिकल संचालन; जैसे—AND, OR, NOT आदि।
- डाटा ट्रांसफर ऑपरेशन (8051 और External Memory के बीच)

'Accumulator' नाम इस तथ्य से आया है कि यह रजिस्टर सभी Arithmatic और अधिकांश Logical फंक्शन के परिणामस्वरूप जमा (या स्टोर) करने के लिए उपयोग किया जाता है।

	0	0	0	0	0	0	0	0	(Value on RESET)
ACC	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Address = E0H
	MSB							LSB	

2. B (Register B)—B रजिस्टर का उपयोग ACC के साथ गुणा और भाग में किया जाता है। ये दो ऑपरेशन डाटा पर किए जाते हैं, जो केवल रजिस्टर्स A और B में संग्रहीत होते हैं गुणन ऑपरेशन के दौरान, ऑपरेंड (गुणक या गुणक) में से एक B रजिस्टर में स्टोर होता है और परिणाम का उच्च बाइट भी होता है। भाग ऑपरेशन में B रजिस्टर में भाजक होता है और शेष परिणाम भी। यह सामान्य संचालन के लिए एक सामान्य उद्देश्य रजिस्टर के रूप में भी उपयोग किया जा सकता है और अक्सर अस्थायी परिणामों को संग्रहीत करने के लिए प्रोग्रामर द्वारा सहायक रजिस्टर के रूप में उपयोग किया जाता है।

रजिस्टर B SFR एड्रेस स्पेस के F0H एड्रेस पर स्थित होता है।

	0	0	0	0	0	0	0	0	(Value on RESET)
B	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Address = F0H
	MSB							LSB	

प्रोग्राम स्टेटस वर्ड Program Status Word (PSW)

PSW या Program Status Word रजिस्टर को फ्लैग रजिस्टर भी कहा जाता है और यह महत्वपूर्ण SFRs में से एक है। PSW रजिस्टर में फ्लैग बिट्स शामिल होती हैं, जो प्रोग्रामर को परिणाम की स्थिति की जांच करने में मदद करते हैं और निर्णय भी लेते हैं। फ्लैग 1-बिट स्टोरेज तत्व है, जो कुछ निर्देशों के निष्पादन से उत्पन्न परिणाम की प्रकृति को संग्रहीत और इंगित करते हैं। निम्न डायग्राम PSW रजिस्टर की कंटेंट को दिखाता है।

	0	0	0	0	0	0	0	0	(Value on RESET)
	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	
PSW	CY	AC	F0	RS1	RS0	OV	–	P	Address = D0H
	MSB							LSB	

निम्न तालिका प्रत्येक फ्लैग के कार्य का वर्णन करती है।

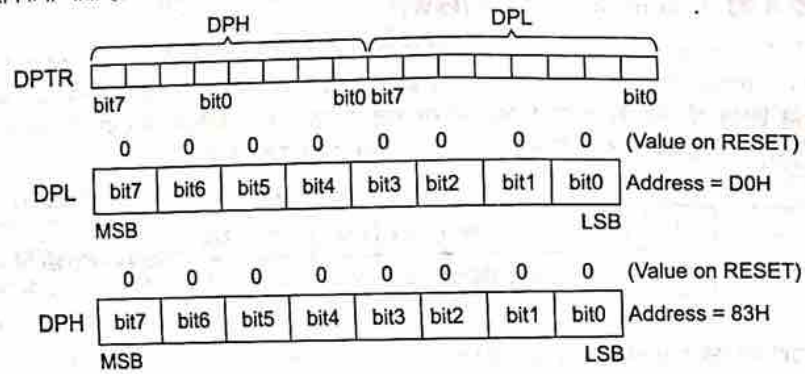
Bit	Symbol	Flag Name			Description
7	C or CY	Carry			Used in Arithmetic, Logic & Boolean Operations
6	AC	Auxiliary Carry			Used in BCD Aritlunetic
5	F0	Flag 0			General Purpose User Flag
4	RS 1	Register Bank Selection Bit 1			
3	RS0	Register Bank Selection Bit 1			
		RS1	RS0	Bank	
		0	0	Bank 0	
		0	1	Bank 1	
		1	0	Bank 2	
		1	1	Bank 3	
2	OV	Overflow			Used in Arithmetic Operations
1	—	Reserved			May be used as a General Purpose Flag
0	P	Parity			Set to 1 if A has odd # of 1's; otherwise Reset

पॉइंटर रजिस्टर्स (Pointer Registers)

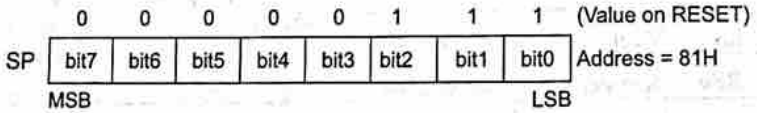
डाटा पॉइंटर Data Pointer (DPTR – DPL and DPH)—डाटा पॉइंटर एक 16-बिट रजिस्टर है और फिजिकल रूप से DPL (डाटा पॉइंटर लो) और DPH (डाटा पॉइंटर हाई) SFR का संयोजन होता है। डाटा पॉइंटर

को 16-बिट रजिस्टर (DPTR के रूप में) या दो 8-बिट रजिस्टर (DPL और DPH) के रूप में उपयोग किया जा सकता है। DPTR का फिजिकल मेमोरी एड्रेस नहीं होता है, लेकिन DPL (DPTR का लोअर बाइट) और DPH (DPTR का उच्च बाइट) SFR मेमोरी स्पेस में अलग-अलग एड्रेस होते हैं। DPL = 82H और DPH = 83H।

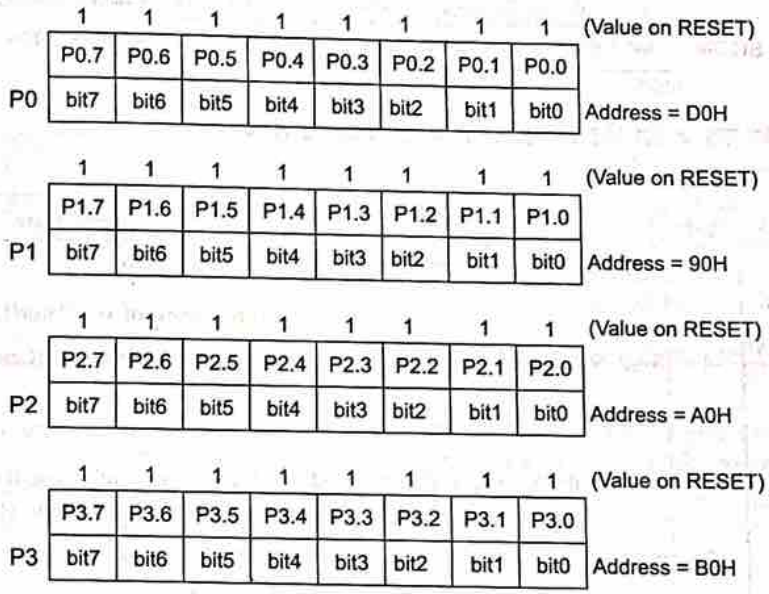
DPTR रजिस्टर का उपयोग External Memory (प्रोग्राम-ROM या डाटा-RAM) को संबोधित करने वाले प्रोग्रामर द्वारा किया जाता है।



स्टैक पॉइंटर Stack Pointer (SP)—SP या Stack Pointer Stack के शीर्ष पर होता है और यह अगले डाटा तक पहुँचने का संकेत देता है। Stack pointer को PUSH, POP, CALL और RET Instructions का उपयोग करके Access किया जा सकता है। स्टैक पॉइंटर 8-बिट रजिस्टर है और रीसेट पर, स्टैक पॉइंटर को 07H के साथ initialized किया जाता है। स्टैक में एक नया डाटा बाइट लिखते समय, SP (स्टैक पॉइंटर) स्वचालित रूप से 1 से बढ़ जाता है और नया डाटा एक पते SP + 1 पर लिखा जाता है। स्टैक से डाटा को पढ़ते समय, डाटा को SP में पते से पुनर्प्राप्त किया जाता है और उसके बाद SP को 1 (एसपी -1) द्वारा घटाया जाता है।



I/O पोर्ट्स रजिस्टर I/O Port Registers (P0, P1, P2 and P3)—8051 माइक्रोकंट्रोलर के चार पोर्ट, जिसका उपयोग Input/Output के रूप में किया जा सकता है। ये चार पोर्ट P0, P1, P2 और P3 हैं। प्रत्येक पोर्ट में एक ही नाम के साथ एक संबंधित रजिस्टर होता है। पोर्ट रजिस्टर्स के पते इस प्रकार हैं: P0 - 80H, P1 - 90H, P2 - A0H और P2 - B0H। इन SFR में प्रत्येक बिट 8051 माइक्रोकंट्रोलर में एक भौतिक पिन से मेल खाती है। ये सभी पोर्ट रजिस्टर बिट एड्रेसेबल और बाइट एड्रेसेबल दोनों हैं। पोर्ट रजिस्टर बिट पर 1 या 0 लिखना संबंधित पिन पर एक उपयुक्त वोल्टेज (5V और 0V) के रूप में प्रतिबिंबित होगा। यदि कोई पोर्ट बिट SET (Declared as 1) है, तो संबंधित पोर्ट पिन को इनपुट के रूप में कॉन्फ़िगर किया जाएगा और इसी प्रकार यदि पोर्ट बिट को स्पष्ट (Declared as 0) किया जाता है, तो संबंधित पोर्ट पिन आउटपुट के रूप में कॉन्फ़िगर किया गया है। रीसेट करने पर, सभी पोर्ट बिट्स सेट (1) हैं और इसलिए, सभी पोर्ट पिन इनपुट के रूप में कॉन्फ़िगर किए गए हैं।

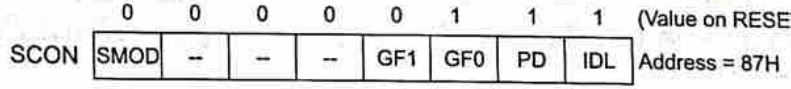


पेरिफेरेल कण्ट्रोल रजिस्टर्स (Peripheral Control Registers)

1. PCON (Power Control)—PCON या पॉवर कंट्रोल रजिस्टर, जैसा कि नाम से पता चलता है, कि इसका उपयोग 8051 माइक्रोकंट्रोलर के पॉवर मोड्स को नियंत्रित करने के लिए किया जाता है। यह SFR मेमोरी स्पेस के 87H पर स्थित है। PCON रजिस्टर में दो बिट्स का उपयोग करके माइक्रोकंट्रोलर को आइडल मोड और पॉवर डाउन मोड में सेट किया जा सकता है। आइडल मोड में माइक्रोकंट्रोलर क्लॉक सिग्नल को ALU (CPU) तक रोक देगा, लेकिन इसे अन्य पेरिफेरेल; जैसे—टाइमर, सीरियल, इंटरप्ट्स आदि को दिया जाता है। आइडल मोड को समाप्त करने के लिए आपको एक इंटरप्ट या हार्डवेयर रीसेट का उपयोग करना होगा। पॉवर डाउन मोड में Oscillator को रोक दिया जाएगा और पॉवर 2 v तक कम हो जाएगी। पावर डाउन मोड को समाप्त करने के लिए आपको हार्डवेयर रीसेट का उपयोग करना होगा।

इन दोनों के अतिरिक्त, PCON रजिस्टर का उपयोग कुछ अतिरिक्त उद्देश्यों के लिए भी किया जा सकता है। PCON रजिस्टर में SMOD बिट का उपयोग सीरियल पोर्ट की बॉड दर को नियंत्रित करने के लिए किया जाता है।

PCON रजिस्टर में दो सामान्य उद्देश्य फ्लैग बिट्स होते हैं, जिनका उपयोग प्रोग्रामर द्वारा निष्पादन के दौरान किया जा सकता है।



SCON (Serial Control)—Serial Control या SCON SFR का उपयोग 8051 माइक्रोकंट्रोलर के Serial Port को नियंत्रित करने के लिए किया जाता है। यह 98H के पते के रूप में स्थित है। SCON का उपयोग करके आप Serial Port के ऑपरेशन मोड को नियंत्रित कर सकते हैं, Serial Port के बॉड रेट और Serial Port का उपयोग करके डाटा भेज सकते या प्राप्त कर सकते हैं।

SCON रजिस्टर में बिट्स भी होते हैं, जो डाटा की बाइट प्रसारित या प्राप्त होने पर स्वचालित रूप से सेट होते हैं।

	0	0	0	0	0	1	1	1	(Value on RESET)
SCON	SM0	SM1	SM2	REN	TB8	RB8	TI	RI	Address = 98H
	MSB							LSB	

सीरियल पोर्ट मोड कंट्रोल बिट (Serial Port Mode Control Bits)

TCON (Timer Control)

SM0	SM1	Mode	Description	Baud Rate
0	0	0	8-Bit Synchronous Shift Register Mode	Fixed Baud Rate (Frequency of oscillator/12)
0	1	1	8-bit Standard UART mode	Variable Baud Rate (Can be set by Timer 1)
1	0	2	9-bit Multi-processor Comm. mode	Fixed Baud Rate (Frequency of oscillator/32 or Frequency of oscillator/64)
1	1	3	9-bit Multi-processor Comm. mode	Variable Baud Rate (Can be set by Timer 1)

Timer Control या TCON रजिस्टर का उपयोग 8051 माइक्रोकंट्रोलर के Timer को शुरू करने या रोकने के लिए किया जाता है। यह इंगित करने के लिए बिट्स भी हैं कि क्या Timer अतिप्रवाह है। TCON SFR में इंटरप्ट संबंधित बिट्स भी होते हैं।

	0	0	0	0	0	0	0	0	(Value on RESET)
SCON	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0	Address = 88H
	MSB							LSB	

TMOD (Timer Mode)—TMOD या Timer Mode रजिस्टर या SFR का उपयोग टाइमर T0 और T1 के ऑपरेटिंग मोड को सेट करने के लिए किया जाता है। निचले चार बिट्स का उपयोग Timer 0 को आकार देने के लिए किया जाता है और Timer 4 को कॉन्फ़िगर करने के लिए उच्चतर बिट्स का उपयोग किया जाता है।

	0	0	0	0	0	0	0	0	(Value on RESET)
TMOD	GATE1	C/T1	T1M1	T1M0	GATE0	C/T0	T0M1	T0M0	Address = 89H
	MSB							LSB	

गेटेक्स बिट का उपयोग INTx पिन के संबंध में या INTx पिन की परवाह किए बिना Timerx को संचालित करने के लिए किया जाता है।

GATE1 = 1 ⇒ Timer1 is operated only if INT1 is SET.

GATE1 = 0 ⇒ Timer1 is operates irrespective of INT1 pin.

GATE0 = 1 ⇒ Timer0 is operated only if INT0 is SET.

GATE0 = 0 ⇒ Timer0 is operates irrespective of INT0 pin.

The C/Tx bit is used selects the source of pulses for the Timer to count.

C/T1 = 1 ⇒ Timer1 counts pulses from Pin T1 (P3.5) (Counter Mode)

C/T1 = 0 ⇒ Timer1 counts pulses from internal oscillator (Timer Mode)

C/T0 = 1 ⇒ Timer0 counts pulses from Pin T0 (P3.4) (Counter Mode)

C/T0 = 0 ⇒ Timer0 counts pulses from internal oscillator (Timer Mode)

T×M0	T×M1	Mode	Description
0	0	0	13-bit Timer Mode (THx – 8-bit and TLx – 5-bit)
0	1	1	16-bit Timer Mode
1	0	2	8-bit Auto Reload Timer Mode
1	1	3	Two 8-bit Timer Mode or Split Timer Mode

Note : x = 0 for Timer 0 and x = 1 for Timer 1.

IE (Interrupt Enable)—IE या Interrupt Enable रजिस्टर का उपयोग व्यक्तिगत Interrupt को सक्षम या अक्षम करने के लिए किया जाता है। यदि कोई बिट SET है, तो संबंधित सक्षम किया जाता है और यदि बिट साफ़ हो जाता है, तो Interrupt अक्षम हो जाता है। IE रजिस्टर के बिट 7 यानी EA बिट का उपयोग सभी Interrupt को सक्षम या अक्षम करने के लिए किया जाता है

	0	0	0	0	0	0	0	0	(Value on RESET)
IE	EA	–	ET2	ES	ET1	EX1	ET0	EX0	Address = A8H
	MSB							LSB	

IP (Interrupt Priority)—IP या Interrupt Priority रजिस्टर का उपयोग Interrupt की प्राथमिकता को उच्च या निम्न रूप में सेट करने के लिए किया जाता है। यदि बिट को CLEARED किया जाता है, तो संबंधित व्यवधान को कम प्राथमिकता दी जाती है और यदि बिट सेट है, तो Interrupt को उच्च प्राथमिकता दी जाती है।

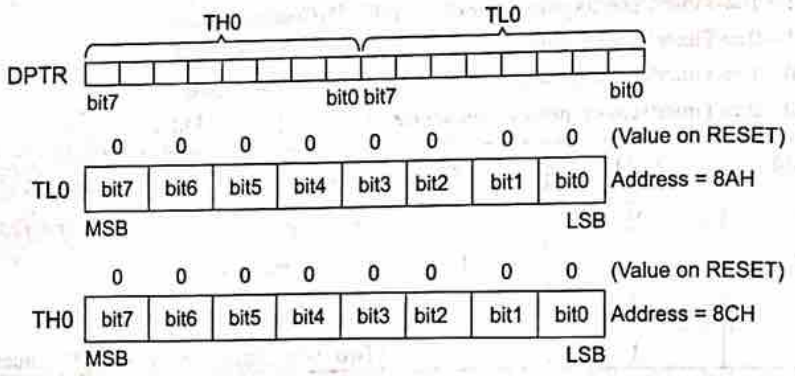
	0	0	0	0	0	0	0	0	(Value on RESET)
IP	–	–	PT2	PS	PT1	PX1	PT0	PX0	Address = 88H
	MSB							LSB	

पेरिफेरेल डाटा रजिस्टर (Peripheral Data Registers)

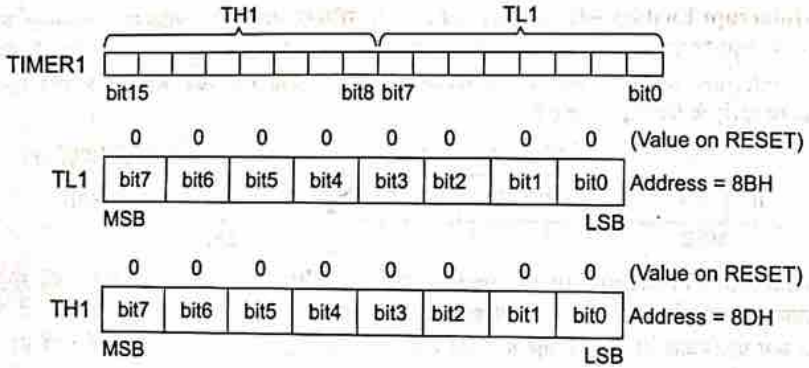
1. SBUF (Serial Data Buffer)—Serial Data Buffer या SBUF रजिस्टर का उपयोग ट्रांसमिशन या रिसेप्शन के दौरान Serial Data को Hold करने के लिए किया जाता है।

	X	X	X	X	X	X	X	X	(Value on RESET)
SBUF	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Address = 99H
	MSB							LSB	

TL0/TH0 (Timer 0 Low/High)—टाइमर 0 में दो SFR होते हैं—TL0 और TH0। TL0 निचली बाइट है और TH0 उच्च बाइट है और साथ में वे एक 16-बिट टाइमर 0 रजिस्टर बनाते हैं।



TL1/TH1 (Timer 1 Low/High)—TL1 और TH1 टाइमर 0 के निचले और उच्च बाइट्स हैं।



8051 माइक्रोकंट्रोलर के साथ External Memory को Interface (Interfacing External Memory with 8051 Micro-controller)

हमने देखा है, कि एक विशिष्ट 8051 माइक्रोकंट्रोलर में 4KB की RAM और 128B RAM और आधुनिक 8051 माइक्रोकंट्रोलर वेरिएंट में 8K ROM और 256B RAM होती है।

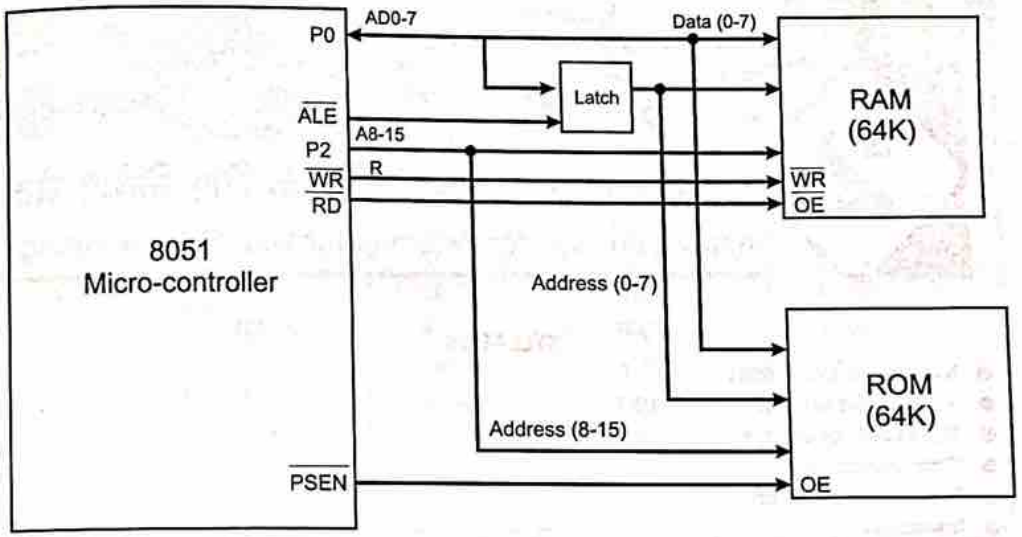
8051 माइक्रोकंट्रोलर पर आधारित प्रणाली का डिजाइनर 8051 माइक्रोकंट्रोलर में उपस्थित Internal RAM और ROM तक सीमित नहीं है। इसमें बाह्य रैम और रोम यानी डाटा मेमोरी और प्रोग्राम दोनों को जोड़ने का प्रावधान है।

बाह्य प्रोग्राम मेमोरी या ROM को इंटरफेयर करने का कारण यह है, कि उच्च स्तरीय भाषाओं में लिखे गए जटिल प्रोग्राम अक्सर बड़े होते हैं और अधिक मेमोरी स्थान घेरते हैं।

एक और महत्वपूर्ण कारण यह है, कि 8031 या 8032 जैसी चिप, जिनमें कोई Internal ROM नहीं है, External RAM के साथ हस्तक्षेप करना होगा।

अधिकतम 64B प्रोग्राम मेमोरी (ROM) और डाटा मेमोरी (RAM) प्रत्येक को 8051 माइक्रोकंट्रोलर के साथ इंटरफेस किया जा सकता है।

निम्न चित्र 8051 माइक्रोकंट्रोलर के साथ External RAM के 64KB और External ROM के 64KB को रोकने के ब्लॉक आरेख को दर्शाता है।



चित्र 1.13 : Micro-control के साथ Externaly Memory को Interface करना

प्रश्नावली

1. माइक्रोप्रोसेसर और माइक्रोकन्ट्रोलर में अन्तर स्पष्ट कीजिए।
2. माइक्रोकन्ट्रोलर के विभिन्न अनुप्रयोगों को बताइये?
3. 8051 माइक्रोकन्ट्रोलर का आरेख बनाकर व्याख्या कीजिए।
4. 8051 माइक्रोकन्ट्रोलर का पिन डायग्राम बनाइए।
5. 8051 माइक्रोकन्ट्रोलर के इनपुट/आउटपुट पोर्ट (I/O पोर्ट) की संरचना बनाकर व्याख्या कीजिए।
6. मैमोरी ऑर्गेनाइजेशन की व्याख्या 8051 माइक्रोकन्ट्रोलर के सन्दर्भ में कीजिए।
7. स्पेशल फंक्शन रजिस्टर (Special Function Registers – SFR) की व्याख्या कीजिए।
8. External मेमोरी बारे में बताइए।
9. PIN PSEN का कार्य 8051 माइक्रोकन्ट्रोलर के सन्दर्भ में समझाइए।
10. 8051 सिस्टम में प्रोग्राम मेमोरी किस प्रकार व्यवस्थित रहती है?

❑

अध्याय

2 माइक्रोकंट्रोलर प्रोग्रामिंग के लिए निर्देश सेट (Instruction Set for Micro-controller Programming)

SYLLABUS

- Instruction Set of 8051
- Addressing Modes,
- Types of Instructions
- Time operation
- Serial Port operation
- Interrupts

2.1 8051 माइक्रोकंट्रोलर इंस्ट्रक्शन सेट का परिचय (Introduction to 8051 Micro-controller Instruction Set)

किसी भी माइक्रोकंट्रोलर के लिए एक प्रोग्राम लिखना एक विशेष क्रम में माइक्रोकंट्रोलर को निर्देश देना होता है जिसमें उन्हें एक विशिष्ट कार्य करने के लिए निष्पादित किया जाता है। माइक्रोकंट्रोलर के निर्देशों को माइक्रोकंट्रोलर के इंस्ट्रक्शन सेट के रूप में जाना जाता है। जिस प्रकार हमारे वाक्य शब्दों से बने होते हैं, उसी प्रकार एक माइक्रोकंट्रोलर प्रोग्राम इंस्ट्रक्शन से बना होता है। एक प्रोग्राम में लिखे गए निर्देश माइक्रोकंट्रोलर को बताते हैं, कि क्या कार्य करना है। निर्देशों का एक सेट कंप्यूटर के लिए यूनिक होता है। 8051 माइक्रोकंट्रोलर इंस्ट्रक्शन सेट को MCS-51 इंस्ट्रक्शन सेट भी कहा जाता है।

जैसा कि माइक्रोकंट्रोलर्स के 8051 परिवार 8-बिट प्रोसेसर हैं, 8051 माइक्रोकंट्रोलर इंस्ट्रक्शन सेट को 8-बिट कंट्रोल एप्लिकेशन के लिए अनुकूलित किया गया है। एक विशिष्ट 8-बिट प्रोसेसर के रूप में, 8051 माइक्रोकंट्रोलर इंस्ट्रक्शन में 8-बिट ओपकोड होता हैं। परिणाम स्वरूप, 8051 माइक्रोकंट्रोलर इंस्ट्रक्शन सेट में $2^8 = 256$ निर्देश हो सकते हैं।

2.2 8051 माइक्रोकंट्रोलर के निर्देश और समूह (8051 Micro-controller Instructions and Groups)

8051 माइक्रोकंट्रोलर इंस्ट्रक्शन सेट, इंस्ट्रक्शन के प्रकार और एड्रेसिंग मोड के विवरण में जाने से पहले, आइए हम 8051 माइक्रोकंट्रोलर इंस्ट्रक्शन सेट (MCS-51 Instruction Set) के निर्देशों और समूहों के बारे में संक्षिप्त रूप से जानते हैं। निम्न तालिका प्रत्येक समूह में 8051 Instruction groups और instruction दिखाती है। 8051 माइक्रोकंट्रोलर निर्देशों के सेट में 49 इंस्ट्रक्शन मेनेमोनिक्स हैं और इन 49 मेनेमिक्स को पांच समूहों में विभाजित किया गया है।

Data Transfer	Arithmetic	Logical	Boolean	Program Branching
MOV	ADD	ANL	CLR	LJMP
MOVC	ADDC	ORL	SETB	AJMP
MOVX	SUBB	XRL	MOV	SJMP
PUSH	INC	CLR	JC	JZ
POP	DEC	CPL	JNC	JNZ
XCH	MUL	RL	JB	CJNE
XCHD	DIV	RLC	JNB	DJNZ
	DA A	RR	JBC	NOP
		RRC	ANL	LCALL
		SWAP	ORL	ACALL
			CPL	RET
				RETI
				JMP

2.3 8051 एड्रेसिंग मोड (8051 Addressing Modes)

एड्रेसिंग मोड एक लक्ष्य डाटा का पता लगाने का एक तरीका है, जिसे ऑपरेंड भी कहते हैं। 8051 फ़ैमिली ऑफ़ माइक्रोकंट्रोलर्स ऑपरेंड्स को संबोधित करने के लिए पाँच प्रकार के एड्रेसिंग मोड्स हैं। ये हैं—

- Immediate Addressing
- Register Addressing
- Direct Addressing
- Register—Indirect Addressing
- Indexed Addressing

1. इमीडियेट एड्रेसिंग (Immediate Addressing)—Immediate Addressing Mode में, Opcode का अनुसरण करने वाला ऑपरेंड, 8 या 16 बिट्स का एक निरंतर डाटा है। Immediate Addressing नाम इस तथ्य से आया है, कि मेमोरी में संग्रहीत किए जाने वाले निरंतर डाटा तुरंत ओपकोड का अनुसरण करते हैं। संग्रहीत की जाने वाली कांस्टेंट वैल्यू को एक रजिस्टर से न लेकर इसे इंस्ट्रक्शन में निर्दिष्ट किया जाता है। गंतव्य रजिस्टर जिसमें कांस्टेंट डाटा को कॉपी किया जाना चाहिए, इंस्ट्रक्शन में उल्लिखित ऑपरेंड के समान होना चाहिए।

Example : MOV A, #030H

यहां, Accumulator 30 (हेक्साडेसिमल) से भरा हुआ है। # ऑपरेंड में यह दर्शाता है, कि यह एक डाटा है न कि किसी रजिस्टर का एड्रेस।

इमीडियेट एड्रेसिंग बहुत तीव्र है, क्योंकि लोड किया जाने वाला डाटा निर्देशों में ही दिया गया है।

2. रजिस्टर एड्रेसिंग (Register Addressing)—8051 माइक्रोकंट्रोलर मेमोरी ऑर्गनाइजेशन में हमने RAM के ऑर्गनाइजेशन और प्रत्येक बैंक में आठ रजिस्टर्स के साथ वर्किंग रजिस्टर्स के चार बैंकों को जाना है। रजिस्टर एड्रेसिंग मोड में आठ रजिस्टर्स में से एक (R0-R7) को निर्देश में ऑपरेंड के रूप में निर्दिष्ट किया जाता है। PSW रजिस्टर की मदद से उपयुक्त बैंक का चयन करना महत्वपूर्ण है। बैंक एड्रेस का चयन करते हुए रजिस्टर एड्रेस का एक उदाहरण देखें।

Example : MOV A, R5

यहां, Bank0 के रजिस्टर R5 की 8-बिट सामग्री को Accumulator पर ले जाया जाता है।

3. डायरेक्ट एड्रेसिंग (Direct Addressing)—डायरेक्ट एड्रेसिंग मोड में निर्देशों को (Instructions) ऑपरेटर के रूप में डाटा का एड्रेस में निर्दिष्ट किया जाता है। डायरेक्ट एड्रेसिंग मोड का उपयोग करके हम किसी भी रजिस्टर या ऑन-चिप वैरिएबल का उपयोग कर सकते हैं। इसमें सामान्य प्रयोजन रैम (General purpose RAM), SFRs, I/O पोर्ट, कंट्रोल रजिस्टर शामिल हैं।

Example : MOV A, 47H

यहाँ, RAM स्थान 47H में डाटा Accumulator पर ले जाया जाता है।

4. रजिस्टर इनडायरेक्ट एड्रेसिंग (Register Indirect Addressing)—Indirect Addressing Mode या रजिस्टर इनडायरेक्ट एड्रेसिंग मोड में, ऑपरेंड का पता रजिस्टर के कंटेंट के रूप में निर्दिष्ट किया जाता है। यह एक उदाहरण के साथ स्पष्ट हो जाएगा।

Example : MOV A, @R1

@ प्रतीक कि अप्रत्यक्ष एड्रेसिंग मोड अप्रत्यक्ष को दर्शाता है। यदि उदाहरण के लिए, R1 की सामग्री 56H है, तो ऑपरेंड आंतरिक RAM स्थान 56H में है। यदि रैम स्थान 56H का कंटेंट 24 H है, तो 24 H को Accumulator में स्थानांतरित किया जाता है।

एड्रेसिंग मोड में केवल R0 और R1 की अनुमति है। इनडायरेक्ट एड्रेसिंग मोड में इन रजिस्टर को पॉइंटर रजिस्टर कहा जाता है।

5. इंडेक्स्ड एड्रेसिंग मोड (Indexed Addressing Mode)—Indexed Addressing मोड में ऑपरेंड का Effective address बेस रजिस्टर और ऑफसेट रजिस्टर का योग होता है। बेस रजिस्टर या तो डाटा पॉइंटर (DPTR) या प्रोग्राम काउंटर (PC) हो सकता है, जबकि ऑफसेट रजिस्टर Accumulator (A) होता है। इंडेक्स्ड एड्रेसिंग मोड में केवल MOVC और JMP निर्देशों का उपयोग किया जा सकता है। इंडेक्स्ड एड्रेसिंग मोड लुक-अप टेबल से डाटा प्राप्त करते समय उपयोगी होता है।

Example : MOVC A, @A+DPTR

यहाँ, ऑपरेंड के लिए पता DPTR और Accumulator की सामग्री का योग है।

2.4 8051 माइक्रोकंट्रोलर इंस्ट्रक्शन सेट में इंस्ट्रक्शन के प्रकार (Types of Instructions in 8051 Micro-controller Instruction Set)

निर्देशों (Instructions) के बारे में जानने से पहले, आइए हम 8051 माइक्रोकंट्रोलर के निर्देशों की संरचना देखते हैं। 8051 निर्देश में एक ओपकोड (Short of Operation – Code) होता है, इसके बाद आकार जीरो बाइट, एक बाइट या दो बाइट्स का ऑपरेंड होता है। निर्देश के अप-कोड भाग में मेंमोनिक शामिल है, जो प्रदर्शन के प्रकार को निर्दिष्ट करता है। निर्देश के सभी Mnemonics या Opcode भाग एक बाइट आकार के होते हैं। निर्देश के ऑपरेंड भाग में आकर यह निर्देशों द्वारा संशोधित किए जा रहे डाटा को परिभाषित करता है। ऑपरैंड निम्नलिखित में से कोई भी हो सकता है—

- No Operand
- Data value
- I/O Port
- Memory Location
- CPU register

कई ऑपरेंड और निर्देश का प्रारूप इस प्रकार है—

2.4.1 Mnemonic Destination Operand, Source Operand

एक साधारण निर्देश में केवल Opcode होता है। अन्य निर्देशों में एक या अधिक ऑपरेंड हो सकते हैं। निर्देश एक-बाइट निर्देश हो सकता है, जिसमें केवल Opcode या दो-बाइट निर्देश होते हैं, जहाँ दूसरा बाइट ऑपरेंड या तीन बाइट निर्देश होता है, जहाँ ऑपरेंड दूसरा और तीसरा बाइट बनाता है। उनके द्वारा किए गए कार्य के आधार पर, 8051 माइक्रोकंट्रोलर के निर्देशों के सेट में सभी निर्देशों को पाँच समूहों में विभाजित किया गया है। जो निम्नलिखित हैं—

- Data Transfer Instructions
- Arithmetic Instructions
- Logical Instructions
- Boolean or Bit Manipulation Instructions
- Program Branching Instructions

1. डाटा ट्रान्सफर निर्देश (Data Transfer Instructions)—डाटा ट्रांसफर निर्देश रजिस्टर्स या बाह्य प्रोग्राम मेमोरी या बाह्य डेटा मेमोरी के बीच डेटा के साथ ट्रांसफर से जुड़े हैं। डाटा ट्रांसफर से जुड़े Mnemonics नीचे दिए गए हैं।

- MOV
- MOVX
- POP
- XCHD
- MOVC
- PUSH
- XCH

निम्न तालिका अन्य संभावित विवरणों के साथ-साथ अन्य विवरण; जैसे—मोड, आकार के कब्जे वाले और मशीन के चक्रों को सूचीबद्ध करती है।

Mnemonic	Instruction	Description	Addressing Mode	# of Bytes	# of Cycles
MOV	A, #Data	A ← Data	Immediate	2	1
	A, Rn	A ← Rn	Register	1	1
	A, Direct	A ← (Direct)	Direct	2	1
	A, @Ri	A ← @Ri	Indirect	1	1
	Rn, #Data	Rn ← data	Immediate	2	1
	Rn, A	Rn ← A	Register	1	1
	Rn, Direct	Rn ← (Direct)	Direct	2	2
	Direct, A	(Direct) ← A	Direct	2	1
	Direct, Rn	(Direct) ← Rn	Direct	2	2
	Direct 1, Direct2	(Direct1) ← (Direct2)	Direct	3	2
	Direct, @Ri	(Direct) ← @R i	Indirect	2	2
	Direct, #Data	(Direct) ← #Data	Direct	3	2
	@Ri, A	@Ri e A	Indirect	1	1

	@Ri, Direct	@Ri ← Direct	Indirect	2	2
	@Ri, #Data	@Ri ← #Data	Indirect	2	1
	DPTR, #Data16	DPTR ← #Data16	Immediate	3	2
MOVC	A, @A+DPTR	A ← Code Pointed by A+DPTR	Indexed	1	2
	A, @A+PC	A ← Code Pointed by A+PC	Indexed	1	2
	A, @Ri	A ← Code Pointed by Ri (8-bit Address)	Indirect	1	2
MOVX	A, @DPTR	A ← External Data Pointed by DPTR	Indirect	1	2
	@Ri, A	@Ri ← A (External Data 8-bit Addr)	Indirect	1	2
	@DPTR, A	@DPTR ← A (External Data 16-bit Addr)	Indirect	1	2
PUSH	Direct	Stack Pointer SP ← (Direct)	Direct	2	2
POP	Direct	(Direct) ← Stack Pointer SP	Direct	2	2
XCH	Rn	Exchange ACC with Rn	Register	1	1
	Direct	Exchange ACC with Direct Byte	Direct	2	1
	@R i	Exchange ACC with Indirect RAM	Indirect	1	1
XCHD	A, @Ri	Exchange ACC with Lower Order Indirect RAM	Indirect	1	1

2. एरिथमेटिक निर्देश (Arithmetic Instructions)—एरिथमेटिक निर्देशों का उपयोग करके आप जोड़, घटा, गुणा और भाग कर सकते हैं। एरिथमेटिक निर्देशों में एक की वृद्धि, एक के द्वारा घटा और एक विशेष निर्देश को Decimal Adjust Accumulator कहा जाता है।

8051 माइक्रोकंट्रोलर इंस्ट्रक्शन सेट के एरिथमेटिक निर्देशों से जुड़े मेमोनिक्स निम्न हैं—

- ADD
- ADDC
- SUBB
- INC
- DEC
- MUL
- DIV
- DA A

एरिथमेटिक निर्देशों में डाटा प्रारूप के बारे में कोई जानकारी नहीं होती है यानी Signed, unsigned, ASCII, BCD, आदि। इसके अलावा, एरिथमेटिक निर्देशों द्वारा किए गए संचालन PSS रजिस्टर में कैरी, ओवरफ्लो, जीरो आदि फ्लैग को प्रभावित करते हैं।

एरिथमेटिक इंस्ट्रक्शन्स से जुड़े सभी संभावित Mnemonics निम्नलिखित तालिका में उल्लिखित हैं।

Mnemonic	Instruction	Description	Addressing Mode	# of Bytes	# of Cycles
ADD	A, #Data	A ← A + Data	Immediate	2	1
	A; Rn	A ← A+Rn	Register	1	1
	A, Direct	A ← A + (Direct)	Direct	2	1
	A, @Ri	A ← A + @Ri	Indirect	1	1
ADDC	A, #Data	A ← A + Data + C	Immediate	2	1
	A, Rn	A← A+Rn+C	Register	1	1
	A. Direct	A ← A+ (Direct) + C	Direct	2	1
	A, @Ri	A ← A + @Ri+C	Indirect	1	1
SUBB	A, #Data	A ← A – Data – C	Immediate	2	1
	A, Rn	A ← A – Rit – C	Register	1	1
	A, Direct	A ← A – (Direct) – C	Direct	2	1
	A, @Ri	A ← A – @Ri – C	Indirect	1	1
MUL	AB	Multiply A with B (A ← Lower Byte of A*13 and B ← Higher Byte of A*B)	—	1	4

DIV	AB	Divide A by B (A ← Quotient and B ← Remainder)	—	1	4
DEC	A	A ← A − 1	Register	1	1
	Rii	Rn ← Rn − 1	Register	1	1
	Direct	(Direct) ← (Direct) − 1	Direct	2	1
	@Ri	@Ri 4 @Ri − 1	Indirect	1	1
INC	A	A ← A + 1	Register	1	1
	Rn	Rn ← Rn + 1	Register	1	1
	Direct	(Direct) ← (Direct) + 1	Direct	2	1
	@Ri	@Ri ← @Ri + 1	Indirect	1	1
	DPTR	DPTR ← DPTR + 1	Register	1	2
DA	A	Decimal Adjust Accumulator	—	1	1

3. तार्किक निर्देश (Logical Instructions)—निर्देशों का अगला समूह लॉजिकल निर्देश हैं, जो लॉजिकल संचालन; जैसे—AND, OR, XOR, NOT, Rotate, Clear और Swap करते हैं। लॉजिकल निर्देश बिट के आधार पर डाटा के बाइट्स पर किए जाते हैं।

लॉजिकल निर्देशों के साथ जुड़े मनोमिक्स निम्नलिखित हैं—

- ANL
- ORL
- XRL
- CLR
- CPL
- RL
- RLC
- RR
- RRC
- SWAP

निम्न तालिका तार्किक निर्देशों के सभी संभावित Mnemonics को दर्शाती है।

Mnemonic	Instruction	Description	Addressing Mode	# of Bytes	# of Cycles
ANL	A, #Data	A ← A AND Data	Immediate	2	1
	A, Rn	A ← A AND Rn	Register	1	1
	A, Direct	A ← A AND (Direct)	Direct	2	1
	A, @Ri	A ← A AND @Ri	Indirect	1	1

	Direct, A	(Direct) ← (Direct) AND A	Direct	2	1
	Direct, #Data	(Direct) ← (Direct) AND #Data	Direct	3	2
ORL	A, #Data	A ← A OR Data	Immediate	2	1
	A, Rn	A ← A OR Rn	Register	1	1
	A, Direct	A ← A OR (Direct)	Direct	2	1
	A, @Ri	A ← A OR @Ri	Indirect	1	1
	Direct, A	(Direct) ← (Direct) OR A	Direct	2	1
	Direct, #Data	(Direct) ← (Direct) OR #Data	Direct	3	2
XRL	A, #Data	A ← A XRL Data	Immediate	2	1
	A, Rn	A ← A XRL Rn	Register	1	1
	A, Direct	A ← A XRL (Direct)	Direct	2	1
	A, @Ri	A ← A XRL @Ri	Indirect	1	1
	Direct, A	(Direct) 4 (Direct) XRL A	Direct	2	1
	Direct, #Data	(Direct) 4 (Direct) XRL #Data	Direct	3	2
CLR	A	A ← 00H	—	1	1
CPL	A	A ← A	—	1	1
RL	A	Rotate ACC Left	—	1	1
RLC	A	Rotate ACC Left through Cany	—	1	1
RR	A	Rotate ACC Right	—	1	1
RRC	A	Rotate ACC Right through Carry	—	1	1
SWAP	A	Swap Nibbles within ACC	—	1	1

4. बूलियन या बिट मैनीपुलेशन निर्देश (Boolean or Bit Manipulation Instructions)—जैसा कि नाम से पता चलता है, बूलियन या बिट मैनिपुलेशन निर्देश बिट चर से सम्बन्धित है । हम जानते हैं, कि RAM में एक विशेष बिट-एड्रेसेबल क्षेत्र है और कुछ विशेष Function Registers (SFRs) भी addressable हैं। बूलियन या बिट मैनिपुलेशन निर्देशों के अनुरूप मेमोनिक्स निम्न हैं—

- CLR
- MOV
- JNC
- JNB
- ANL
- CPL
- SETB
- JC
- JB
- JBC
- ORL

ये निर्देश बिट स्तर पर सेट, स्पष्ट और, या, पूरक आदि प्रदर्शन कर सकते हैं। बूलियन निर्देशों के सभी संभावित Mnemonics निम्नलिखित तालिका में निर्दिष्ट किए गए हैं।

Mnemonic	Instruction	Description	# of Bytes	# of Cycles
CLR	C	C ← 0 (C = Cany Bit)	1	1
	Bit	Bit e 0 (Bit = Direct Bit)	2	1
SET	C	C ← 1	1	1
	Bit	Bit ← 1	2	1
CPL	C	C ← C	1	1
	Bit	Bit ← Bit	2	1
ANL	C, /Bit	C ← C. Bit (AND)	2	1
	C, Bit	C ← C. Bit (AND)	2	1
ORL	C, /Bit	C ← C+ $\overline{\text{Bit}}$ (OR)	2	1
	C, Bit	C ← C+ Bit (OR)	2	1
MOV	C, Bit	C ← Bit	2	1
	Bit, C	Bit ← C	2	2

JC	rel	Jump is Carry (C) is Set	2	2
JNC	rel	Jump is Carry (C) is Not Set	2	2
JB	Bit, rel	Jump is Direct Bit is Set	3	2
JNB	Bit, rel	Jump is Direct Bit is Not Set	3	2
JBC	Bit, tel	Jump is Direct Bit is Set and Clear Bit	3	2

5. प्रोग्राम ब्रांचिंग निर्देश (Program Branching Instructions)—8051 माइक्रोकंट्रोलर निर्देशों में निर्देशों का अंतिम समूह प्रोग्राम ब्रांचिंग निर्देश हैं। ये निर्देश प्रोग्राम लॉजिक के प्रवाह को नियंत्रित करते हैं। प्रोग्राम ब्रांचिंग इंस्ट्रक्शंस के Mnemonics इस प्रकार हैं।

- LJMP
- SJMP
- JNZ
- DJNZ
- LCALL
- RET
- JMP
- AJMP
- JZ
- CJNE
- NOP
- ACALL
- RETI

NOP (No Operation) को छोड़कर ये सभी निर्देश प्रोग्राम काउंटर (PC) को एक या दूसरे तरीके से प्रभावित करते हैं। इनमें से कुछ निर्देशों में कार्यक्रम के अन्य भाग पर नियंत्रण स्थानांतरित करने से पहले निर्णय लेने की क्षमता होती है।

निम्न तालिका कार्यक्रम के निर्देश के संबंध में सभी वर्णव्यवस्था को दिखाती है।

Mnemonic	Instruction	Description	# of Bytes	# of Cycles
ACALL	ADDR 1 1	Absolute Subroutine Call PC + 2 → (SP); ADDRI 1 → PC	9	9
LCALL	ADDR 16	Long Subroutine Call PC + 3 → (SP); ADDR16 → PC	3	1
RET	—	Return from Subroutine (SP) 3 PC	1	9
RETI	—	Return from Interrupt	1	2
AJMP	ADDR 11	Absolute Jump ADDR11 → PC	2	2
LJMP	ADDR16	Long Jump ADDR16 3 PC	3	2
SJMP	rel	Short Jump PC + 2 + rel 3 PC	2	2

JMP	@A + DPTR	A + DPTR → PC	1	2
JZ	rel	If A = 0, Jump to PC + rel	2	2
JNZ	rel	If A ≠ 0, Jump to PC + rel		
CJNE	A, Direct, rel	Compare (Direct) with A. Jump to PC + rel if not equal	3	2
	A, #Data, rel	Compare #Data with A. Jump to PC + rel if not equal	3	2
	Rn, #Data, rel	Compare #Data with Rn. Jump to PC + rel if not equal	3	2
	@Ri, #Data, rel	Compare #Data with @Ri. Jump to PC + rel if not equal	3	2
DJNZ	Rn, rel	Decrement Rn. Jump to PC + rel if not zero	2	2
	Direct, rel	Decrement (Direct). Jump to PC + rel if not zero	3	2
NOP		No Operation	1	1

2.5 टाइमर ऑपरेशन (Timer Operation)

INTEL 8051 में दो 16-बिट टाइमर रजिस्टर होते हैं। इन रजिस्टर्स को टाइमर 0 और टाइमर 1 के नाम से जाना जाता है। टाइमर रजिस्टर को दो मोड में उपयोग किया जा सकता है। ये मोड हैं टाइमर मोड और काउंटर मोड। इन दो मोडों के बीच एकमात्र अंतर टाइमर रजिस्टर्स को बढ़ाने के लिए स्रोत है।

1. टाइमर मोड (Timer Mode)—टाइमर मोड में आंतरिक मशीन चक्र को गिना जाता है। इसलिए यह रजिस्टर प्रत्येक मशीन चक्र में बढ़ जाता है। इसलिए जब क्लॉक की आवृत्ति 12 मेगाहट्र्ज होती है, तो प्रत्येक मिली सेकंड में टाइमर रजिस्टर बढ़ जाता है। इस मोड में यह बाहरी टाइमर इनपुट पिन को अनदेखा करता है।

2. काउंटर मोड (Counter Mode)—काउंटर मोड में बाहरी घटनाओं को गिना जाता है। इस मोड में बाहरी इनपुट पिन के प्रत्येक 1 से 0 ट्रांजीशन के लिए टाइमर रजिस्टर बढ़ जाता है। इस प्रकार के ट्रांजीशन को घटनाओं के रूप में माना जाता है। बाहरी इनपुट पिन को प्रत्येक मशीन चक्र में एक बार नमूना लिया जाता है, और 1or 0 ट्रांजीशन का निर्धारण करने के लिए एक और मशीन चक्र की आवश्यकता होगी। इस मोड में कम-से-कम दो मशीन चक्रों की आवश्यकता होती है। जब आवृत्ति 12MHz है, तो अधिकतम गणना आवृत्ति 12MHz / 24 = 500KHz होगी। तो घटना की गणना के लिए समय अवधि 2μs होती है।

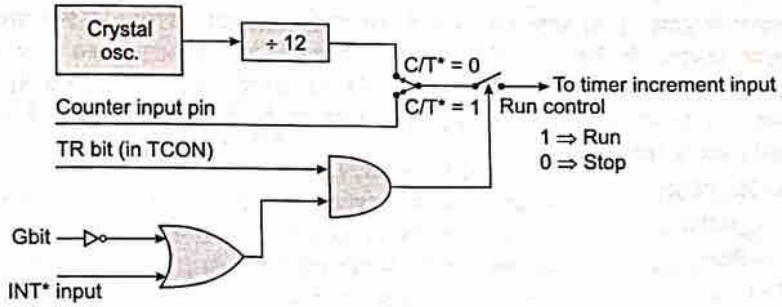
टाइमर या काउंटर के चार अलग-अलग मोड हैं। मोड 0 से मोड 2 टाइमर/काउंटर दोनों के लिए हैं। प्रत्येक टाइमर रजिस्टर के लिए मोड 3 का एक अलग अर्थ है। TMOD रजिस्टर को इन टाइमर या काउंटर को कॉन्फ़िगर

करने के लिए प्रोग्राम किया जा सकता है। सीरियल पोर्ट का उपयोग मोड 1 और 3 में सीरियल संचार के लिए किया जाता है। बॉडर रेट उत्पन्न करने के लिए टाइमर 1 का उपयोग किया जाता है। टाइमर या काउंटर ऑपरेशन के लिए केवल टाइमर 0 उपलब्ध होता है।

TMOD Register—TMOD (टाइमर मोड) एक SFR है। इस रजिस्टर का एड्रेस 89H होता है। यह बिट-एड्रेसेबल नहीं होता है—

Timer	Timer1 Mode				Timer0 Mode			
Bit Details	Gate (G)	C/T	M1	M0	Gate (G)	C/T	M1	M0

अब, हम सर्किट को देखते हैं, जो टाइमर के चलने को नियंत्रित करता है—



चित्र 2.1 : टाइमर ऑपरेशन

निम्न तालिका में, हम उच्च या निम्न मूल्य के लिए बिट विवरण और उनके विभिन्न कार्यों को जानेंगे।

Bit Details	High Value(1)	Low Value(0)
C/T	Configure for the Counter operations	Configure for the Timer operations
Gate (G)	Timer0 or Timer1 will be in RunMode when TRX bit of TCON register is high.	Timer0 or Timer1 will be in RunMode when TRX bit of TCON register is high and INT0 or INT1 is high.

Bit Details	00	01	10	11
M1 M0	This is for Mode 0. (8-bit timer/counter, with 5-bit pre-scaler)	This is Mode 1. (16-bit timer/counter)	This is Mode 3 (8-bit auto reload-timer/ counter)	This is Mode 3 (The function depends on Timer0 or Timer1)

जब टाइमर या काउंटर मोड 0 से 2 के बीच में होगा तो गेट बिट अधिक होगा।

उदहारण—Timer0 को 16-बिट इवेंट काउंटर और Timer1 को 8-बिट Auto-reload Counter के रूप में कॉन्फ़िगर करने के लिए, हम बिट पैटर्न 0 0 1 1 0 0 1 0 0 1 का उपयोग कर सकते हैं। यह 25H के बराबर है। यदि हम इस बिट पैटर्न के साथ TMOD रजिस्टर करना चाहते हैं, तो हम इस निर्देश का उपयोग कर सकते हैं—

MOVTMOD, #25H

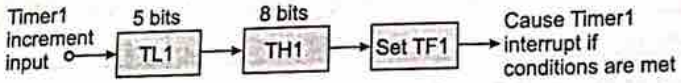
उपरोक्त निर्देश को निष्पादित किया जाता है, फिर टाइमर / काउंटर को सॉफ्टवेयर द्वारा नियंत्रित किया जाएगा। सिस्टम को हार्डवेयर नियंत्रित मोड के रूप में कॉन्फ़िगर करने के लिए, फिर गेट बिट्स होंगे। 1. बिट पैटर्न 1 0 1 0 1 1 0 0=1H होगा

हम इस निर्देश का उपयोग कर सकते हैं—

MOVTMOD, #0ADH

2.5.1 टाइमर/काउंटर मोड 0 (Mode 0 of Timer/Counter)

0 ऑपरेशन में 8-बिट टाइमर या 5-बिट पूर्व-स्केलर के साथ काउंटर होता है। तो यह एक 13-बिट टाइमर/काउंटर है। यह TL0 या TL1 के 5 बिट्स और TH0 या TH1 के 8-बिट्स का उपयोग करता है।



चित्र 2.2 : टाइमर मोड 0

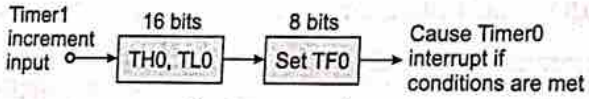
इस उदाहरण में टाइमर 1 का चयन किया गया है, इस मामले में काउंटर ऑपरेशन के लिए प्रत्येक 32 (25) इवेंट्स या टाइमर ऑपरेशन के लिए 32 मशीन चक्र, TH1 रजिस्टर 1 से बढ़ जाएगा। जब F1H से 00H तक TH1overflows, तब TF1 TCON रजिस्टर उच्च होगा, और यह टाइमर / काउंटर को रोकता है। इसलिए एक उदाहरण के लिए, हम कह सकते हैं, कि यदि TH1 F0H को पकड़े हुए है, और यह टाइमर मोड में है, तो TF1 10H * 32 = 512 मशीन चक्र के बाद उच्च होगा।

MOVTMOD, #00H
MOVTH1, #0F0H
MOVIE, #88H
SETB TR1

उपरोक्त कार्यक्रम में टाइमर 1 को टाइमर मोड के रूप में कॉन्फ़िगर किया गया है। इसमें गेट = 0, तब F1H के साथ TH1 को लोड किया जाएगा, फिर टाइमर 1 बाधा को सक्षम करें। अंतिम बार TCON रजिस्टर के TR1 को सेट करें, और टाइमर शुरू करता है।

2.5.2 टाइमर/काउंटर मोड 1 (Mode 1 of Timer/Counter)

मोड 1 ऑपरेशन में 16-बिट टाइमर या काउंटर होता है। निम्नलिखित चित्र में हम टाइमर 1 के लिए मोड 1 का उपयोग कर रहे हैं।



चित्र 2.3 : टाइमर मोड 1

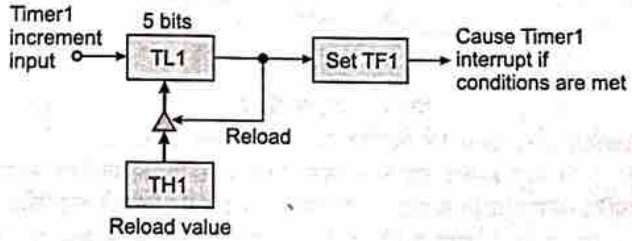
इस स्थिति में टाइमर ऑपरेशन के लिए काउंटर ऑपरेशन या मशीन चक्र के लिए हर घटना, TH0— TL रजिस्टर-जोड़ी 1 से बढ़ाई जाएगी। जब FFFFH से 0000H तक रजिस्टर जोड़ी ओवरफ्लो हो जाती है, तो TCON रजिस्टर का TF0 उच्च होगा, और यह टाइमर / काउंटर को बंद कर देता है। एक उदाहरण के लिए, हम कह सकते हैं, कि यदि TH0 - TL0 रजिस्टर जोड़ी FFF0H को पकड़े हुए है, और यह टाइमर मोड में है, तो TF0 10H = 16 मशीन चक्र के बाद उच्च होगा। जब क्लॉक की आवृत्ति 12 मेगाहर्ट्ज होती है, तो निम्नलिखित निर्देश टाइमर 0 के चलने के बाद एक Interrupt 16 μs उत्पन्न करते हैं।

MOVTMOD, #01H
MOVTL0, #0F0H
MOVTH0, #0FFH
MOVIE, #82H
SETB TR0

उपरोक्त प्रोग्राम में टाइमर 0 को टाइमर मोड के रूप में कॉन्फ़िगर किया गया है। इस मामले में गेट = 0. तब TL0 को F0H के साथ लोड किया जाएगा और TH0 को FFH के साथ लोड किया जाएगा, फिर टाइमर 0 को सक्षम करेंगा। अंतिम बार TCON रजिस्टर का TR0 सेट करें, और टाइमर शुरू करें।

2.5.3 टाइमर /काउंटर मोड 2 (Mode 2 of Timer/Counter)

मोड 2 ऑपरेशन 8-बिट Auto Reload Timer या काउंटर है। दिए गए चित्र में हम टाइमर 1 के लिए मोड 2 का उपयोग कर रहे हैं।



चित्र 2.4 : टाइमर मोड 2

इस स्थिति में टाइमर ऑपरेशन के लिए काउंटर ऑपरेशन या मशीन चक्र के लिए हर घटना, TL1 register 1 से बढ़ जाएगा। जब FFH से 00H तक रजिस्टर्स ओवरफ्लो हो जाते हैं, तो TCON रजिस्टर का TF1 उच्च होगा, TT1 को भी पुन: लोड किया जाएगा। TH1 की सामग्री और फिर से ऑपरेशन शुरू करता है। इसलिए एक उदाहरण के लिए, हम कह सकते हैं, कि यदि TH1 और TL1 रजिस्टर दोनों में F0H है और यह टाइमर मोड में है, तो TF1 10H = 16 मशीन चक्र के बाद उच्च होगा। जब क्लॉक की आवृत्ति 12MHz होती है, तो यह 16 μs के बाद होती है, तो निम्नलिखित निर्देश टाइमर 1 के चलने के बाद हर 16μs के बाद एक बार एक अवरोध उत्पन्न करते हैं।

MOVTMOD,#20h
MOVTL1, #0F0H
MOVTH1, #0F0H
MOVITE, #88H
SETBTR1

उपरोक्त प्रोग्राम में टाइमर 1 को टाइमर मोड के रूप में कॉन्फ़िगर किया गया है। इस मामले में गेट = 0. तब TL1 और TH1 को F0H के साथ लोड किया गया है। इसके बाद Timer1 इंटरप्ट को सक्षम करें। अंतिम बार TCON रजिस्टर के TR1 को सेट करें, और टाइमर शुरू करें।

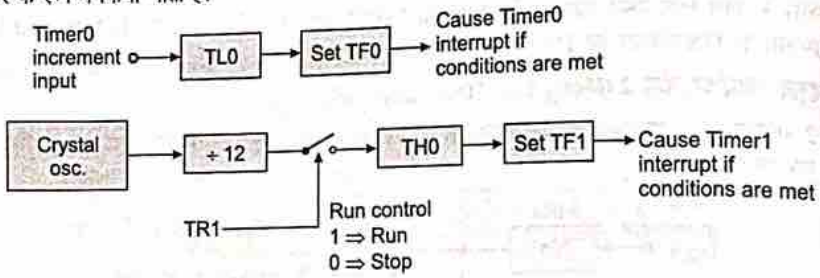
मोड 2 में टाइमर 1 वांछित बॉड दर उत्पन्न करता है, जब सीरियल पोर्ट मोड 1 या 3 पर काम कर रहा होता है।

2.5.4 टाइमर /काउंटर मोड 3 (Mode 3 of Timer/Counter)

टाइमर 0 और टाइमर 1 के लिए मोड 3 अलग है। जब टाइमर 0 मोड 3 में कार्य करता है, तो TL0 का उपयोग 8-बिट टाइमर / काउंटर के रूप में किया जाता है। यह मानक Timer0 नियंत्रण बिट्स, T0 और INT0 इनपुट द्वारा नियंत्रित किया जाएगा। TH0 का उपयोग 8-बिट टाइमर के रूप में किया जाता है, लेकिन काउंटर पर नहीं। यह टाइमर 1 कंट्रोल बिट TR1 द्वारा नियंत्रित होता है। जब F0H से 00H तक TH0 ओवरफ्लो होता है, तब TF1 को 1 पर सेट किया जाता है। निम्नलिखित चित्र में, हम मोड 3 में Timer0 कर सकते हैं।

जब टाइमर 1 मोड 3 में कार्य करता है, तो यह केवल गिनती रखता है, लेकिन चलता नहीं है। जब Timer0 मोड 3 में है, Timer1 को मोड 0, 1 और 2 में से एक में कॉन्फ़िगर किया जाता है। इस स्थिति में Timer 1 माइक्रोकंट्रोलर को बाधित नहीं कर सकता है। जब TF 1 का उपयोग TH0 Timer द्वारा किया जाता है, तो Timer 1 का उपयोग बॉड

रेट जनरेटर के रूप में किया जाता है।



चित्र 2.5 : टाइमर मोड 3

मोड 3 के लिए Timer 0 और Timer 1 में गेट बिट का अर्थ इस प्रकार है—

यह मोड 0, 1, या 2 की तरह 8-बिट टाइमर / काउंटर TL0 के रनिंग को नियंत्रित करता है। TH0 का रनिंग केवल TR1 बिट द्वारा नियंत्रित किया जाता है। तो टाइमर 0 के लिए इस मोड में गेट बिट की कोई विशेष भूमिका नहीं है।

मोड 3 अतिरिक्त 8-बिट टाइमर / काउंटर की आवश्यकता वाले अनुप्रयोगों के लिए उपस्थित है। Timer 0 के मोड 3 में, 8051 में तीन टाइमर हैं। TH0 द्वारा एक 8-बिट टाइमर, TL0 द्वारा एक और 8-बिट टाइमर / काउंटर, और Timer1 द्वारा एक 16-बिट टाइमर / काउंटर।

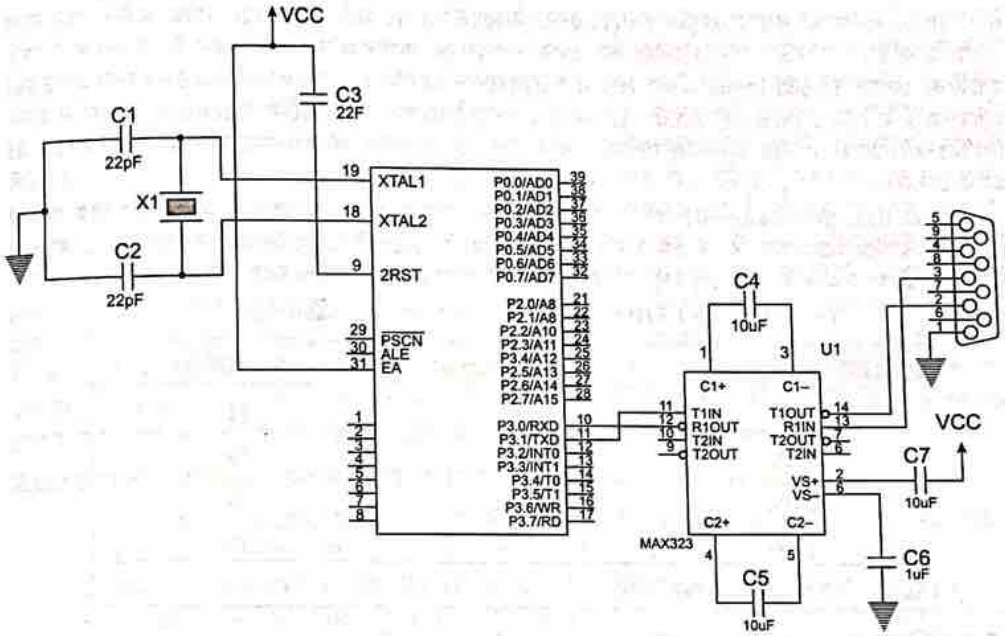
यदि Timer0 मोड 3 में है, और Timer1 0, 1 या 2 पर कार्य कर रहा है, तो गेट बिट कम या INT1 अधिक होने पर Timer1 का नियंत्रण सक्रिय होता है। गेट उच्च होने और INT1 कम होने पर रन कंट्रोल निष्क्रिय हो जाता है।

2.6 8051 में शृंखला संचार (Serial Communication in 8051)

माइक्रोकंट्रोलर्स को संचार के लिए डाटा एकत्र करने के लिए सेंसर, कंप्यूटर और इतने पर बाह्य उपकरणों के साथ संवाद करने की आवश्यकता होती है। डाटा संचार मुख्य रूप से दो तरीकों से किया जाता है—समानांतर और सीरियल मोड। समानांतर मोड में डाटा बिट्स को अधिक डाटा पिन का उपयोग करके तेजी से स्थानांतरित किया जाता है। लेकिन जब एक माइक्रोकंट्रोलर की बात आती है, तो हम डाटा ट्रांसफर के लिए कई पिन समर्पित करने का जोखिम नहीं उठा सकते। 8051 माइक्रोकंट्रोलर में UART या सीरियल संचार नियंत्रक को केवल दो पिन का उपयोग करके डाटा भेजने और प्राप्त करने की अनुमति देता है। सीरियल कम्युनिकेशन माइक्रोकंट्रोलर और बाह्य उपकरणों के बीच संचार स्थापित करने के लिए केवल दो डाटा पिन का उपयोग करता है। संचार डाटा के इस मोड में एक समय में एक बिट को स्थानांतरित किया जाता है। यह आलेख अपने सीरियल पोर्ट RS232 के माध्यम से संचार स्थापित करने के लिए पीसी के साथ 8051 के इंटरफेसेस का वर्णन करता है।

2.6.1 असेंबली में सीरियल पोर्ट प्रोग्रामिंग (Serial Port Programming in Assembly)

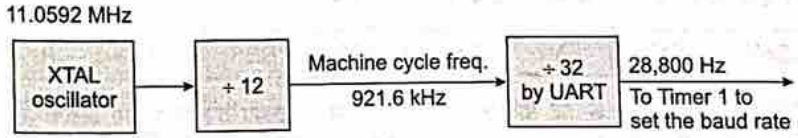
चूंकि *IBM PC/compatible* कंप्यूटर 8051 पर आधारित प्रणालियों के साथ संचार करने के लिए व्यापक रूप से उपयोग किए जाते हैं, PC के COM पोर्ट के साथ 8051 के सीरियल संचार पर जोर दिया जाएगा। डाटा की अनुमति देने के लिए PC और 8051 सिस्टम के बीच बिना किसी त्रुटि के स्थानांतरण हमें यह सुनिश्चित करना चाहिए कि 8051 सिस्टम की बॉड दर PC के COM पोर्ट की बॉड दर से मेल खाती है।



चित्र 2.6 : असेम्बली में सीरीयल पोर्ट प्रोग्रामिंग

2.6.2 8051 में बॉड रेट (Baud Rate in 8051)

8051 स्थानान्तरण करता है और कई अलग-अलग बॉड दरों पर क्रमिक रूप से डाटा प्राप्त करता है। 8051 के सीरियल संचार को COM पोर्ट के माध्यम से PC के साथ स्थापित किया गया है। यह सुनिश्चित करना चाहिए कि 8051 सिस्टम की बॉड दर PC के COM पोर्ट/किसी भी सिस्टम को इंटरफेस करने की बॉड दर से मेल खाती है। 8051 में बॉड दर प्रोग्राम करने योग्य है। यह टाइमर की मदद से किया जाता है। जब सीरियल पोर्ट के लिए उपयोग किया जाता है, तो टाइमर की आवृत्ति (XTAL / 12) / 32 द्वारा निर्धारित की जाती है और प्रत्येक टाइमर अवधि (टाइमर शुरू करने से टाइमर समाप्ति की समय अवधि) के लिए 1 बिट प्रसारित की जाती है। 8051 में क्रिस्टल आवृत्ति और बॉड दर के बीच संबंध यह है कि 8051 मशीन चक्र आवृत्ति प्राप्त करने के लिए क्रिस्टल आवृत्ति को 12 से विभाजित करता है, जो कि चित्र 2.7 में दिखाया गया है। यहाँ XTAL = 11.0592 मेगाहट्र्ज है, मशीन चक्र की आवृत्ति 921.6 kHz है। 8051 के UART बॉड दर को सेट करने के लिए टाइमर 1 द्वारा उपयोग किए जाने से पहले 921.6 kHz की मशीन चक्र आवृत्ति को एक बार और 32 से विभाजित करता है। 32 से विभाजित 921.6 kHz 28,800 हट्र्ज देता है। टाइमर 1 को मोड 2 में प्रोग्राम किया जाना चाहिए, जो कि 8-बिट, ऑटो-रीलोड है।



चित्र 2.7 : बॉड रेट

सीरियल संचार में यदि डाटा 9600 की बॉड दर और XTAL के उपयोग के साथ स्थानांतरित किया जाता है, तो 11.0592 है, उसके बाद TH1 मान को लोड करने के लिए खोजने के लिए निम्नलिखित चरणों का पालन किया जाता

है— टाइमर क्लॉक की क्लॉक आवृत्ति: f = (11.0592 मेगाहर्ट्ज / 12) / 32 = 28,800Hz प्रत्येक क्लॉक की समय अवधि टिक: $T0 = 1/f = 1/28800$ टाइमर की अवधि: $n * T0$ (n) क्लॉक की टिक की संख्या है) 9600 बॉड → 1 प्रतीक की अवधि: 1/9600 $1/9600 = n * T0 = n * 1/28800$ $n = f/9600 = 28800/9600 = 3 \rightarrow TH1 = -3$ इसी प्रकार, बॉड के लिए 2400 $n = f/2400 = 12 \rightarrow TH1 = -12$ उदाहरण: 9600 MOV TMOD, # 20H पर बॉड दर निर्धारित करें; टाइमर 1, मोड 2 (ऑटो रीलोड) MOV TH1, # -3; 9600 बॉड दर सेट करने के लिए सेट TR 1; प्रारंभ टाइमर 1

Baud Rate Selection—बॉड दर टाइमर 1 द्वारा चुना जाता है और जब टाइमर 1 का उपयोग बॉड दर को सेट करने के लिए किया जाता है, तो इसे 2 मोड में 8-बिट, ऑटो-रीलोड में प्रोग्राम किया जाना चाहिए। PC के साथ संगत बॉड दरें प्राप्त करने के लिए, हमें TH1 को तालिका 1 में दिखाए गए मानों के साथ लोड करना होगा।

Table. 1 Timer 1 THI register values for different baud rates

Baud Rate	TH1 (Decimal)	TH1 (Hex)
9600	–3	FD
4800	–6	FA
2400	–12	F4
1200	–24	F8
Note : XTAL = 11.0592 MHz.		

2.6.3 सीरियल संचार के लिए रजिस्टर्स (Registers for Serial Communication)

SBUF (Serial Buffer) Register—यह 8051 में सीरियल संचार के लिए पूर्ण रूप से उपयोग किया गया एक 8 बिट रजिस्टर है। TxD लाइन के माध्यम से स्थानांतरित किए जाने वाले डाटा के बाइट को SBUF रजिस्टर में रखा जाना चाहिए। जब RxD लाइन प्राप्त होती है, तो SBUF डेटा का बाइट धारण करता है। इसे किसी भी अन्य रजिस्टर की तरह एक्सेस किया जा सकता है—

```
MOV SBUF, #'D' ; load SBUF=44H, ASCII for 'D,,
MOV SBUF, A ; copy accumulator into SBUF
MOV A, SBUF ; copy SBUF into accumulator
```

जब एक बाइट लिखी जाती है, तो इसे Start and Stop बिट्स के साथ तैयार किया जाता है और TxD पिन के माध्यम से क्रमिक रूप से स्थानांतरित किया जाता है। जब बिट्स को RxD के माध्यम से क्रमिक रूप से प्राप्त किया जाता है, तो Stop and Start बिट्स को हटाकर, प्राप्त आंकड़ों से एक बाइट बनाकर, और फिर इसे SBUF में रख दिया जाता है।

SCON (Serial Control) Register—यह एक 8 बिट रजिस्टर है, जिसका उपयोग प्रोग्राम को Start Bit, Stop Bit, और डाटा बिटिंग के डाटा बिट्स को अन्य चीजों के बीच करने के लिए किया जाता है।

SM0	SM1	SM2	REN	TB8	RB8	TI	RI

पहले दो बिट्स *SM0, SM1* हैं, जो सीरियल पोर्ट मोड बिट्स हैं। इनका उपयोग फ्रेमिंग प्रारूप को निर्दिष्ट करने के लिए किया जाता है, बॉड की गणना कैसे करें। उदाहरण के लिए यदि (SM0, SM1) = (0,1), मोड 1: 8-बिट डाटा, 1 स्टार्ट बिट, 1 स्टॉप बिट, वेरिएबल बॉड दर टाइमर द्वारा निर्धारित किया जा सकता है। अन्य तीन मोड का उपयोग शायद ही कभी किया जाता है और वे हैं (SM0, SM1) = (0,0), मोड 0: फिक्स्ड बॉड = XTAL / 12, (SM0, SM1) = (1,0), मोड 2 : 9-बिट डाटा, फिक्स्ड बॉड, (SM0, SM1) = (1, 1), मोड 3: 9-बिट डाटा, वेरिएबल बॉड। तीसरे बिट का उपयोग संचार के लिए उपयोग किए जाने वाले प्रोसेसर के प्रकार का चयन करने के लिए किया जाता है। यदि SM2 0 है, तो यह एकल प्रोसेसर संचार है। यदि SM2 1 है, तो यह मल्टीप्रोसेसर संचार है। चौथा बिट REN रिसीव इनेबल है, जिसका उपयोग रिसेप्शन को सक्षम / अक्षम करने के लिए किया जाता है। यदि REN = 1 है, तब 8051 सीरियल पोर्ट से आने वाले डाटा को स्वीकार करेगा। यदि REN = 0 है, तो रिसीवर अक्षम है। जैसे SETB REN, CLR REN, SETB SCON.4, CLR SCON.4। पांचवां बिट TB 8 है जो 8-बिट ट्रांसमिशन के लिए मोड 2 और 3 द्वारा उपयोग किया जाता है। जब मोड 1 का उपयोग किया जाता है, तो पिन TB8 को साफ कर दिया जाना चाहिए। छठे बिट RB 8 का उपयोग बिट के रिसेप्शन के लिए मोड 2 और 3 द्वारा किया जाता है। स्टॉप बिट को स्टोर करने के लिए मोड 1 द्वारा इसका उपयोग किया जाता है। सातवाँ बिट TI है, जो ट्रांसमिट इंटरप्ट है। जब 8051 8-बिट चरित्र के हस्तांतरण को पूरा करता है, तो यह इंगित करने के लिए TI को '1' में सेट करता है, कि यह अगले चरित्र को स्थानांतरित करने के लिए तैयार है। TI को स्टॉप बिट की शुरुआत में उठाया जाता है। अंतिम बिट RI है, जो प्राप्त रुकावट है। जब 8051 को एक पात्र मिलता है, तो UART स्टार्ट बिट को हटा देता है और बिट को रोक देता है। UART SBUF में 8-बिट चरित्र डालता है। RI को 1 R पर सेट किया जाता है, यह इंगित करने के लिए कि SBUF में एक नई बाइट लेने के लिए तैयार है। स्टॉप बिट के माध्यम से आधे रास्ते में उठाया जाता है

2.6.3 डाटा को समानानतर क्रम में भेजने के चरण (Steps to send data serially)

1. सेट बॉड की दर 20H मान के साथ TMOD लोड करके, यह बॉड दर सेट करने के लिए मोड 2 (8-बिट ऑटो-रीलोड) में टाइमर 1 का संकेत देता है
2. सीरियल डाटा ट्रांसफर के लिए बॉड दर निर्धारित करने के लिए उचित मूल्यों के साथ TH1 भरी हुई है
3. SCON रजिस्टर को 50H मान के साथ लोड किया गया है, जो सीरियल मोड 1 को दर्शाता है, जहाँ 8-बिट डाटा को Start/Stop बिट्स के साथ बनाया गया है
4. TR1 को टाइमर 1 शुरू करने के लिए 1 पर सेट किया गया है
5. TI को CLR TI निर्देश द्वारा मंजूरी दे दी जाती है।
6. चरित्र बाइट को क्रमिक रूप से स्थानांतरित करने के लिए SBUF रजिस्टर में लिखा जाता है।
7. निर्देश को पूरी तरह से स्थानांतरित कर दिया गया है या नहीं यह देखने के लिए निर्देश JNB TI, xx के उपयोग के साथ TI ध्वज बिट की निगरानी की जाती है।
8. अगले बाइट को स्थानांतरित करने के लिए, चरण 5 पर जाएँ।

2.6.4 Program to Transfer letter 'D' Serially at 9800 Baud, Continuously

```
MOV TMOD,#20H ; timer 1,mode 2(auto reload)
MOV TH1, #-3 ; 9600 baud rate
MOV SCON, #50H ; 8-bit, 1 stop, REN enabled
SETB TR1 ; start timer 1
AGAIN: MOV SBUF, #'D' ; letter 'D' to transfer
HERE: JNB TI, HERE ; wait for the last bit
CLR TI ;clear TI for next char
SJMP AGAIN ; keep sending A
```

2.6.5 TI फ्लैग का महत्व (Importance of the TI Flag)

TI फ्लैग बिट की जाँच करें, हम जानते हैं, कि 8051 किसी अन्य बाइट को स्थानांतरित करने के लिए तैयार है या नहीं। डाटा के हस्तांतरण के बाद TI Flag बिट 8051 तक उठाया जाता है। 'CLRTI' जैसे निर्देश द्वारा TI Flag

प्रोग्रामर द्वारा clear किया जाता है। SBUF में बाइट लिखते समय, TI Flag थोड़ा उठाए जाने से पहले, यह बाइट के एक हिस्से को हस्तांतरित होने का नुकसान हो सकता है।

2.6.6 Steps to Receive Data Serially

1. मूल्य 20H के साथ TMOD रजिस्टर लोड करके बॉड दर निर्धारित करें, यह बॉड दर सेट करने के लिए मोड 2 (8-बिट ऑटो-रीलोड) में टाइमर 1 का संकेत देता है।
2. बॉड दर निर्धारित करने के लिए उचित मूल्यों के साथ TH1 भरी हुई है।
3. SCON रजिस्टर को 50H मान के साथ लोड किया गया है, जो सीरियल मोड 1 को दर्शाता है, जहां 8-बिट डाटा को Start/Stop बिट्स के साथ बनाया गया है।
4. TR1 को टाइमर 1 शुरू करने के लिए 1 पर सेट किया गया है।
5. RI को CLR RI के निर्देश द्वारा साफ़ किया जाता है।
6. RI Flag बिट की निगरानी JNB RI के उपयोग के साथ की जाती है, यह देखने के लिए कि क्या पूरा चरित्र अभी तक प्राप्त हुआ है या नहीं।
7. जब RI उठाया जाता है, तो SBUF की बाइट होती है; इसकी सामग्री को एक सुरक्षित स्थान पर ले जाया जाता है
8. अगला चरित्र प्राप्त करने के लिए, चरण 5 पर जाएँ।

2.6.7 Program to Receive Bytes of Data Serially, and Put Them in P2, set the Baud Rate at 9600, 8-bit data, and 1 stop bit :

```
MOV TMOD, #20H ; timer 1,mode 2(auto reload)
MOV TH1, #-3 ; 9600 baud rate
MOV SCON, #50H ; 8-bit, 1 stop, REN enabled
SETB TR1 ; start timer 1
HERE: JNB RI, HERE ; wait for char to come in
MOV A, SBUF ; saving incoming byte in A
MOV P2, A ; send to port 1
CLR RI ; get ready to receive next byte
SJMP HERE ; keep getting data
```

2.7 8051 माइक्रोकंट्रोलर में व्यवधान (Interrupts in 8051 Micro-controller)

8051 माइक्रोकंट्रोलर में सबसे शक्तिशाली और महत्वपूर्ण विशेषताएँ व्यवधान हैं। अधिकांश वास्तविक समय की प्रक्रियाओं में, कुछ स्थितियों को ठीक से संभालने के लिए, वास्तविक कार्य को कुछ समय के लिए रोक दिया जाना चाहिए - इसमें आवश्यक कार्रवाई होती है - और फिर मुख्य कार्य पर वापस लौटना चाहिए। इस प्रकार के कार्यक्रमों को निष्पादित करने के लिए, व्यवधान आवश्यक हैं। यह पूर्ण रूप से मतदान पद्धति से भिन्न होता है, जिसमें प्रोसेसर को क्रमिक रूप से प्रत्येक डिवाइस की जाँच करनी चाहिए और यह पूछना चाहिए, कि अधिक प्रोसेसर समय का उपभोग करते समय सेवा की आवश्यकता है या नहीं। 8051 माइक्रोकंट्रोलर में हस्तक्षेप करने वाले उपकरणों या इनबिल्ट डिवाइसिज की नियमित स्थिति की जाँच को कम करने के लिए अधिक वांछनीय हैं। व्यवधन एक ऐसी घटना है, जो मुख्य कार्यक्रम को अस्थायी रूप से निलंबित कर देती है, एक विशेष कोड अनुभाग पर नियंत्रण को पारित करती है, घटना से संबंधित कार्यों को निष्पादित करती है और मुख्य कार्यक्रम के प्रवाह को फिर से शुरू करती है जहाँ इसे छोड़ दिया था।

व्यवधान विभिन्न प्रकार के होते हैं; जैसे—सॉफ्टवेयर और हार्डवेयर, maskable and non-maskable, फिक्स्ड और वेक्टर व्यवधान। व्यवधान सर्विस रूटीन (ISR) व्यवधान होने पर तस्वीर में आता है, और फिर प्रोसेसर को व्यवधान के लिए उचित कार्रवाई करने के लिए कहता है, और ISR के निष्पादन के बाद नियंत्रक मुख्य प्रोग्राम में कूद जाता है।

2.7.1 8051 माइक्रोकंट्रोलर में व्यवधान के प्रकार (Types of Interrupts in 8051 Micro-controller)

8051 माइक्रोकंट्रोलर पाँच अलग-अलग घटनाओं को पहचान सकता है, जो मुख्य कार्यक्रम को सामान्य निष्पादन से व्यवधान करने का कारण बनता है। 8051 में व्यवधान के पाँच स्रोत निम्नलिखित है—

1. Timer 0 overflow interrupt- TF0
2. Timer 1 overflow interrupt- TF1
3. External hardware interrupt- INT0
4. External hardware interrupt- INT1
5. Serial communication interrupt- RI/TI

टाइमर और सीरियल व्यवधान को आंतरिक रूप से माइक्रोकंट्रोलर द्वारा उत्पन्न किया जाता है, जबकि बाहरी व्यवधान को अतिरिक्त इंटरफेसिंग डिवाइस या स्विच द्वारा उत्पन्न किया जाता है जो बाहरी रूप से माइक्रोकंट्रोलर से जुड़े होते हैं। इन बाहरी व्यवधान को बढ़त ट्रिगर या स्तर ट्रिगर किया जा सकता है। जब कोई व्यवधान उत्पन्न होता है, तो माइक्रोकंट्रोलर व्यवधान सर्विस रूटीन को निष्पादित करता है ताकि मेमोरी लोकेशन उस इंटरएक्टिव से मेल खाती है जो इसे सक्षम करता है। मेमोरी लोकेशन के लिए व्यवधान नीचे दी गई व्यवधान वेक्टर टेबल में दिया गया है।

Interrupt	Flag	Interrupt vector address
Reset	–	0000H
INT0 (Ext. int. 0)	IE0	0003H
Timer 0	TF0	000BH
INT1 (Ext. int. 1)	IE1	0013H
Timer 1	TF1	001BH
Serial	TI/RI	0023H

Reset—यह सबसे अधिक प्राथमिकता वाला व्यवधान है, 0x0000 एड्रेस से 8051 माइक्रोकंट्रोलर कोड को निष्पादित करना (Executing) शुरू करने पर Reset करता है।

Internal Interrupt (Timer Interrupt)—8051 में दो आंतरिक व्यवधान हैं जिनका नाम Timer 0 और Timer 1 है। जब भी Timer Outerflow होता है, Timer Overflow Flag (TF0 / TF1) सेट होते हैं। तब माइक्रोकंट्रोलर अपने वेक्टर एड्रेस पर जाकर व्यवधान उत्पन्न करता है। इसके लिए, वैश्विक और Timer व्यवधान को इनेबल किया जाना चाहिए।

Serial Interrupt—8051 में एक सीरियल कम्युनिकेशन पोर्ट होता है और संबंधित सीरियल इंटरप्ट फ्लैग (TI / RI) होता है। जब एक बाइट का अंतिम बिट (स्टॉप बिट) प्रसारित होता है, तो TI सीरियल इंटरप्ट फ्लैग को सेट किया जाता है, और जब प्राप्त डाटा बाइट का अंतिम बिट (स्टॉप बिट) प्राप्त होता है, तो RI फ्लैग सेट हो जाता है।

IE Register : Interrupt Enable Register—IE रजिस्टर का उपयोग व्यवधान स्रोतों को enable/disable करने के लिए किया जाता है।

7	6	5	4	3	2	1	0	
EA	—	—	ES	ET1	EX1	ET0	EX0	IE

Bit 7 – EA : Enable All Bit
- **1** = Enable all interrupts
- **0** = Disable all interrupts

Bit 6,5 – Reserved bits

Bit 4 – ES : Enable Serial Interrupt Bit
- **1** = Enable serial interrupt
- **0** = Disable serial interrupt

Bit 3 – ET1 : Enable Timer1 Interrupt Bit
- **1** = Enable Timer1 interrupt
- **0** = Disable Timer1 interrupt

Bit 2 – EX1: Enable External1 Interrupt Bit
- **1** = Enable External1 interrupt
- **0** = Disable External1 interrupt

Bit 1 – ET0 : Enable Timer0 Interrupt Bit
- **1** = Enable Timer0 interrupt
- **0** = Disable Timer0 interrupt

Bit 0 – EX0 : Enable External0 Interrupt Bit
- **1** = Enable External0 interrupt
- **0** = Disable External0 interrupt

Interrupt Priority : व्यवधान को प्राथमिकता रजिस्टर (IP) का उपयोग करके सौंपा जा सकता है—
Priority to the interrupt can be assigned by using the **Interrupt Priority Register (IP)**

Interrupt Priority after Reset :

Priority	Interrupt source	Intr. bit / flag
1	External Interrupt 0	INT0
2	Timer Interrupt 0	TF0
3	External Interrupt 1	INT1
4	Timer Interrupt 1	TF1
5	Serial interrupt	(TI/RI)

तालिका में, Reset पर प्राथमिकताओं को इंटरप्ट किया गया है। 8051 में व्यवधान की प्राथमिकताओं के अनुसार, जब तक माइक्रोकंट्रोलर को उच्च प्राथमिकता वाले कार्यक्रम के साथ समाप्त नहीं किया जाता है, तब तक सबसे कम प्राथमिकता वाले अंतराल को सेवा नहीं दी जाती है। ऐसे मामले में जब दो या दो से अधिक व्यवधान प्राथमिकता के अनुसार माइक्रोकंट्रोलर की कतार में पहुँचते हैं।

IP Register (Interrupt Priority Register)—8051 में व्यवधान को प्राथमिकता देने के लिए एक इंटरप्ट प्राथमिकता रजिस्टर है।

7	6	5	4	3	2	1	0	
—	—	—	PS	PT1	PX1	PT0	PX0	IP

Bit 7,6,5 – Reserved bits.

Bit 4 – PS: Serial Interrupt Priority Bit
- **1** = Assign a high priority to serial interrupt.
- **0** = Assign low priority to serial interrupt.

Bit 3 – PT1 : Timer1 Interrupt Priority Bit
- **1** = Assign high priority to Timer1 interrupt.
- **0** = Assign low priority to Timer1 interrupt.

Bit 2 – PX1 : External Interrupt 1 Priority Bit
- **1** = Assign high priority to External1 interrupt.
- **0** = Assign low priority to External1 interrupt.

Bit 1 – PT0 : Timer0 Interrupt Priority Bit
- **1** = Assign high priority to Timer0 interrupt.
- **0** = Assign low priority to Timer0 interrupt.

Bit 0 – PX0 : External0 Interrupt Priority Bit
- **1** = Assign high priority to External0 interrupt.
- **0** = Assign low priority to External0 interrupt.

8051 में बाह्य व्यवधान (External interrupts in 8051)

- 8051 में दो बाह्य व्यवधान INT0 और INT1 हैं।
- बाह्य व्यवधान PORT3.2, PORT3.3 पर level or edge प्रदान करके, बाह्य व्यवधान से 8051 नियंत्रक को व्यवधान किया जा सकता है।
- बाह्य पेरिफेरेल इन बाह्य व्यवधान के माध्यम से माइक्रोकंट्रोलर को व्यवधान कर सकती है यदि वैश्विक और बाह्य व्यवधान सक्षम हैं।
- फिर माइक्रोकंट्रोलर वर्तमान निर्देश को निष्पादित करेगा और व्यवधान करने के लिए इंटरप्ट सर्विस रूटीन (ISR) पर जम्प कर जाएगा।
- पोलिंग में विधि को पिन की निगरानी करके एक पल्स के लिए माइक्रोकंट्रोलर की निरन्तर जाँच करनी पड़ती है, जबकि, व्यवधान विधि में माइक्रोकंट्रोलर को मतदान करने की आवश्यकता नहीं होती है। जब भी कोई व्यवधान होता है तो माइक्रोकंट्रोलर व्यवधान अनुरोध का कार्य करता है।

बाह्य व्यवधान के दो प्रकार के सक्रियण स्तर हैं।

1. Edge triggered (Interrupt occur on rising/falling edge detection)
2. Level triggered (Interrupt occur on high/low-level detection)

8051 में, दो प्रकार के सक्रियण स्तर का उपयोग किया जाता है। ये हैं,

Low Level Triggered : जब भी INT0 / INT1 पिन पर एक निम्न स्तर का पता लगाया जाता है, जबकि

वैश्विक और बाह्य व्यवधान सक्षम होते हैं, तो नियंत्रक सेवा (ISR) को बाधित करने के लिए सर्विस रूट को इंटरप्ट करता है।

Falling Edge Triggered—जब भी Falling edge को INT0 / INT1 पिन पर पता लगाया जाता है, जबकि वैश्विक और बाह्य व्यवधान को सक्षम किया गया है, नियंत्रक Service Routine (ISR) को बाधित करने के लिए सेवा को इंटरप्ट करता है।

बाह्य व्यवधान प्रकार और ISR स्थिति का चयन और निगरानी करने के लिए आवश्यक TCON रजिस्टर में निचले चार फ्लैग बिट्स होते हैं।

TCON: Timer/ counter Register

7	6	5	4	3	2	1	0
TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

Bit 3- IE1—बाह्य व्यवधान 1 edge फ्लैग, हार्डवेयर द्वारा सेट किया जाता है, जब INT1 पिन पर व्यवधान होता है और व्यवधान होने पर हार्डवेयर द्वारा क्लियर होता है।

Bit 2- IT1—यह बिट INT1 पिन पर बाहरी इंटरप्ट घटना प्रकार का चयन करता है,

1 = sets interrupt on falling edge

0 = sets interrupt on low level

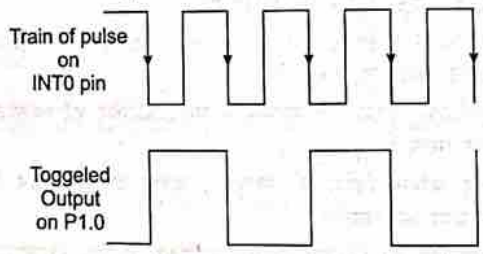
Bit 1- IE0—व्यवधान 0 edge फ्लैग, हार्डवेयर द्वारा सेट INT0 पिन पर व्यवधान होता है और व्यवधान होने पर हार्डवेयर द्वारा क्लियर किया जाता है।

Bit 0 - IT0—यह बिट INT0 पिन पर बाहरी व्यवधान घटना के प्रकार का चयन करता है।

1 = sets interrupt on falling edge

0 = sets interrupt on low level

Example : मान लो कि AT89C51 का बाह्य व्यवधान ऐसा है, कि जब गिरने का निशान INT0 पिन पर पता चलता है, तो माइक्रोकंट्रोलर P1.0 पिन को चालू कर देगा।



चित्र 2.7

Programming Steps :

1. Enable global interrupt *i.e.* EA = 1
2. Enable external interrupt *i.e.* EX0 = 1
3. Enable interrupt trigger mode *i.e.* whether interrupt is edge triggered or level triggered, here we will use falling edge trigger interrupt, so make IT0 = 1.

Program :

```
/*
 * 8051_External_Interrupt
 * http://www.electronicwings.com
 */

#include <reg51.h>     /* Include x51 header file */
sbit LED = P1 ^ 0;     /* set LED on port1 */

void Ext_int_Init()
{
    EA = 1;          /* Enable global interrupt */
    EX0 = 1;         /* Enable Ext. interrupt0 */
    IT0 = 1;         /* Select Ext. interrupt0 on falling edge */
}

void External0_ISR() interrupt 0
{
    LED = ~LED;   /* Toggle pin on falling edge on INT0 pin */
}

void main()
{
    Ext_int_Init();    /* Call Ext. interrupt initialize */
    while(1);
}
```

Note : For level triggered interrupt IT0 needs to be cleared *i.e.* IT0 = 0.

प्रश्नावली

1. माइक्रोकन्ट्रोलर के सन्दर्भ में विभिन्न एड्रसिंग मोड (Addressing mode) की व्याख्या उदाहरण सहित कीजिए।
2. विभिन्न प्रकार के निर्देशों के बारे में बताइए।
3. माइक्रोकन्ट्रोलर में टाइमर ऑपरेशन किस प्रकार होता है?
4. माइक्रोकन्ट्रोलर में सीरियल पोर्ट ऑपरेशन की व्याख्या कीजिए।
5. माइक्रोकन्ट्रोलर में कितने प्रकार के व्यवधान होते हैं? इनकी विस्तृत व्याख्या कीजिए।
6. 8051 के निर्देशों की संक्षिप्त व्याख्या कीजिए।
7. 8051 में टाइमर के कार्य और ऑपरेटिंग मोड की व्याख्या कीजिए।

8. 8051 सीरियल पोर्ट के Separating mode '0', mode 2 एवं mode 3 की व्याख्या कीजिए।
9. 8051 में CALL statement की व्याख्या कीजिए।
10. 8051 में JUMP statement की व्याख्या कीजिए।
11. 8051 में दो संख्याओं को जोड़ने का प्रोग्राम लिखिए।
12. 30H Accumulator, DPH और DPL में Load करने के लिए 8051 हेतु प्रोग्राम लिखिए।
13. सबरूटीन से आप क्या समझते हैं?
14. SWAP का क्या कार्य है?
15. डिबगिंग से क्या तात्पर्य है?
16. 8051 माइक्रोकन्ट्रोलर के Arithmetic operations का वर्णन कीजिए। उदाहरण द्वारा यह भी समझाइए कि FLAG किस प्रकार प्रभावित होते हैं?
17. निम्न Addressing mode की व्याख्या कीजिए—
    (a) Immidate (b) Register (c) Register Indirct (d) Direct.
18. 8051 के लिए निम्नलिखित Instructions की व्याख्या कीजिए—
    (a) SETB 86 H (c) SETB
    (b) CLR 87 H (d) 0A7H
19. TCON register का उपयोग बताइए।
20. TCON register में क्या लोड़ करना चाहिए? यदि Timer 0 और Timer 1 को शुरू करना है?
21. 8051 के विभिन्न व्यवधान की व्याख्या कीजिए। Highest priority interrupt कौन-सा होता है?
22. Micro-controller का कौन-सा Port, bit addressible है?
23. SCON register के लिए 8 bit के कार्य लिखिए।
24. PCON और ISCON Register के उपयोग बताइए।
25. 8051 Internal और External memory में अन्तर किस प्रकार करता है?

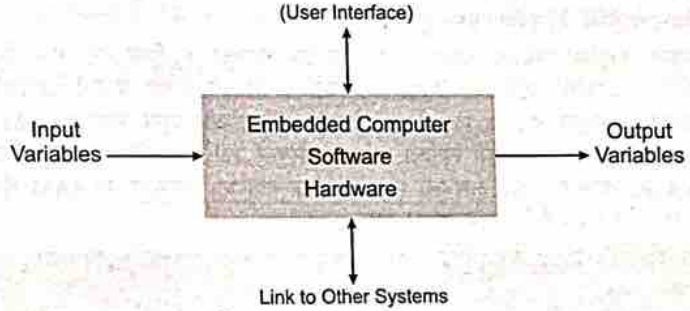
एम्बेडेड सिस्टम का परिचय
(Introduction to Embedded System)

SYLLABUS

Introduction, history of embedded system, embedded system architecture, functional structure of embedded system.

3.1 परिचय (Introduction)

एम्बेडेड सिस्टम एक ऐसा Computer सिस्टम है, जिसे एक विशिष्ट कार्य (Task) को करने के लिए बनाया जाता है। एम्बेडेड सिस्टम सॉफ्टवेर तथा हार्डवेयर का एक संयोजन (Combination) होता है। एम्बेडेड का अर्थ है, एक वस्तु का दूसरी वस्तु से जुड़ना। तो इस प्रकार हम कह सकते है, कि एम्बेडेड सिस्टम एक कम्प्यूटर हार्डवेयर सिस्टम है, जिसमें सॉफ्टवेर जुड़ा हुआ होता है। एम्बेडेड सिस्टम एक स्वतंत्र सिस्टम हो सकता है या फिर यह एक बहुत बड़े सिस्टम का एक भाग हो सकता है।



चित्र 3.1 : Embedded System

एक एम्बेडेड सिस्टम Microcontroller या Microprocessor पर आधारित सिस्टम होता है और इसे किसी विशेष कार्य को करने के लिए बनाया जाता है। उदाहरण एक fire alarm system एक एम्बेडेड सिस्टम है, जो केवल धुएँ को ही सेंस कर सकता है, एम्बेडेड सिस्टम एक इलेक्ट्रॉनिक सिस्टम है, जिसमें एक सॉफ्टवेयर होता है और वह सॉफ्टवेयर कम्प्यूटर हार्डवेयर में एम्बेडेड होता है। एक एम्बेडेड सिस्टम एप्लिकेशन के आधार पर Programmable या Non-programmable होता है । एम्बेडेड सिस्टम को नियमों के अनुसार एक या एक से अधिक कार्यों को करने, व्यवस्थित करने, या कार्य करने के तरीके के रूप में परिभाषित किया गया है। एम्बेडेड सिस्टम में, सभी इकाइयाँ एक साथ एकत्रित होती हैं और कार्यक्रम के अनुसार कार्य करती हैं। एम्बेडेड सिस्टम के उदाहरणों में माइक्रोवेव ओवन, वॉशिंग मशीन, प्रिंटर, ऑटोमोबाइल, कैमरा आदि जैसे कई उपकरण शामिल हैं। ये सिस्टम माइक्रोप्रोसेसर, माइक्रोकंट्रोलर के साथ-साथ डीएसपी जैसे प्रोसेसर का उपयोग करते हैं।

एम्बेडेड सिस्टम की महत्वपूर्ण विशेषताएँ में गति, आकार, शक्ति, विश्वसनीयता, सटीकता, अनुकूलनशीलता आदि शामिल हैं। इसलिए, जब एम्बेडेड सिस्टम उच्च गति से कार्य करता है, तो इसका उपयोग वास्तविक समय के कार्यों के

लिए किया जा सकता है, आपकी जानकारी के लिए बता दे की इसमें सिस्टम का आकार और विद्युत की खपत बहुत कम होनी चाहिए, तभी आप सिस्टम को विभिन्न स्थितियों के लिए आसानी से अनुकूल कर सकते है।

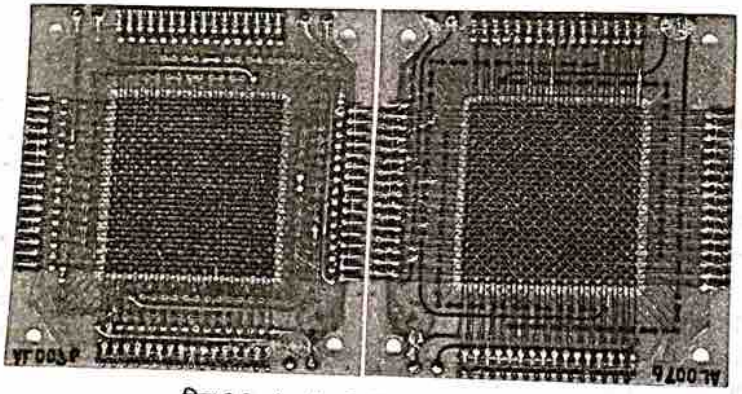
एम्बेडेड सिस्टम को एक बड़े इलैक्ट्रिकल सिस्टम के साथ 'एम्बेडेड' किया जाता हैं और सामान्य उपयोगी कम्प्यूटर जैसे पर्सनल कम्प्यूटर (PC) की विभिन्न प्रकार की आवश्यकताओं को पूरा करने के लिए इसे डिज़ाइन किया जाता है। एम्बेडेड सिस्टम आज विभिन्न प्रकार के बिजली वाले उपकरणों में पाए जा सकते हैं, जैसे—मोबाइल फोन, औद्योगिक मशीन, ऑटोमोबाइल, कैमरा, घरेलू उपकरण, हवाई-जहाज, वैल्डिंग मशीन और खिलौने के साथ-साथ ईसीजी रिकॉर्डर जैसे चिकित्सा उपकरणों में भी आप एम्बेडेड सिस्टम को देख सकते है। एम्बेडेड सिस्टम आमतौर पर माइक्रो कंट्रोलर पर आधारित होते हैं। माइक्रोप्रोसेसर और माइक्रो कंट्रोलर के बीच में एक छोटा सा अन्तर होता है, वो यह कि माइक्रोप्रोसेसर में केवल सीपीयू होता है—इनमें इनबिल्ट रैम या रोम नहीं होती है, इसलिए इन्हें बाह्य रूप से जोड़ने की आवश्यकता होती है, माइक्रो कंट्रोलर इस मायने में बेहतर होते हैं, क्योंकि उनमें सीपीयू के साथ-साथ एक निश्चित मात्रा में रैम और रोम भी होती हैं।

एम्बेडेड सिस्टम में तीन कंपोनेंट्स हार्डवेयर, सॉफ्टवेयर और एक Real Time Operating System (RTOS) होता है, जो कि एप्लिकेशन सॉफ़्टवेयर को निगरानी करता है और प्रोसेसर को एक प्रोसेस करने का mechanism (तंत्र) प्रदान करता है। RTOS यह निश्चित करता है, कि सिस्टम कार्य किस प्रकार करेगा। यह एप्लिकेशन प्रोग्राम को Execute करने के दौरान नियमों को निर्धारित करता है। एक छोटे स्तर के एम्बेडेड सिस्टम में RTOS नहीं होता है।

3.1 एम्बेडेड सिस्टम का इतिहास (History of Embedded System)

सन् 1930-40 में कम्प्यूटर का उपयोग केवल एक कार्य के लिए ही किया जाता था और ये आकार में बहुत बड़े होते थे, जो कि आज के समय में एम्बेडेड कम्प्यूटर सिस्टम की तुलना में काफी महंगे होते थे फिर धीरे-धीरे कम्प्यूटर टेक्नोलॉजी में प्रोग्रामेबल कंट्रोलर का उपयोग होना शुरू हुआ।

इसमें सबसे पहले आधुनिक एम्बेडेड सिस्टम अपोलो गाइडेंस कम्प्यूटर था, जिसे सन् 1960 के दशक के आरम्भ में MIT प्रयोगशाला इंस्ट्रूमेंटेशन में चार्ल्स स्टार्क ड्रेपर द्वारा विकसित किया गया था। अपोलो गाइडेंस कम्प्यूटर (AGC) एक ऑन-बोर्ड डिजिटल कम्प्यूटर था, जो कि कमांड मोड्यूल (CM) और लूनर मॉड्यूल (LM) दोनों में अपोलो एस्क्राफ्ट प्रोग्राम पर स्थापित किया गया था। अपोलो फ्लाइट कम्प्यूटर इंटीग्रेटेड सर्किट (IC) का उपयोग करने वाला पहला कम्प्यूटर था। AGC सॉफ्टवेर AGC असेंबली भाषा में लिखा गया था। कम्प्यूटर की RAM चुंबकीय कोर मेमोरी (4K शब्द) और ROM कोर रोप मेमोरी (32K शब्द) थी।



चित्र 3.2 : Apollo Guidence Computer

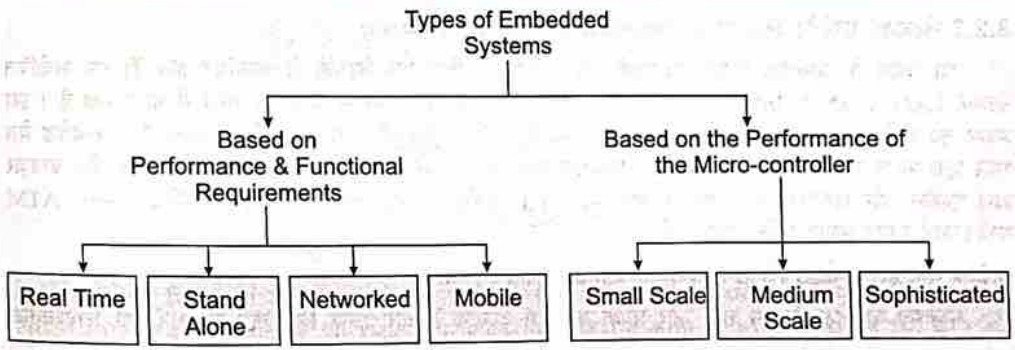
पहला बड़े पैमाने पर उत्पादित एम्बेडेड सिस्टम Minuteman मिसाइल के लिए विकसित किया गया। यह असतत ट्रांजिस्टर लॉजिक (Discrete transistor logic) से निर्मित था और इसमें मुख्य मेमोरी के लिए एक हार्ड डिस्क थी। जब सन् 1966 में Minuteman II का उत्पादन शुरू हुआ, तो D-17 पहला कम्प्यूटर हुआ जिसमे बड़ी संख्या में इंटीग्रेटेड सर्किट का उपयोग हुआ था।

पहला माइक्रोप्रोसेसर, जिसमे बाह्य मेमोरी थी, जिसका उपयोग कैलकुलेटर और अन्य छोटी प्रणालियों में किया गया था, वह 8-बिट माइक्रोप्रोसेसर, 8008, 8080, 8085, 6800, 6502, Z-8० थे। Apple कम्पनी द्वारा 6502 का उपयोग Apple-II को डिजाइन करने के लिए किया गया था, जिसने पर्सनल कम्प्यूटर युग को जन्म दिया।

एम्बेडेड सिस्टम का इतिहास कुछ इस प्रकार हैं—

- 1960 में, MIT में चार्ल्स स्टार्क ड्रेपर द्वारा अपोलो गाइडेंस सिस्टम विकसित करने के लिए पहली बार एम्बेडेड सिस्टम का प्रयोग किया गया था ।
- सन् 1965 में, ऑटोनेटिक्स ने डी-7B का विकसित किया, इस कम्प्यूटर का उपयोग मिनुटमैन मिसाइल मार्गदर्शन प्रणाली में किया गया था।
- सन् 1968 में, एक वाहन के लिए पहला एम्बेडेड सिस्टम बनाया गया था।
- टेक्सास इंस्ट्रूमेंट्स (Texas Instruments) ने सन् 1971 में पहला माइक्रोकंट्रोलर विकसित किया था ।
- सन् 1987 में, विंड एम्बेडेड द्वारा पहला एम्बेडेड OS, VxWorks विकसित किया गया था।
- सन् 1996 में Microsoft ने पहला विंडोज एम्बेडेड CE जारी किया था।
- सन् 1990 के दशक के अंत तक, पहला एम्बेडेड लिनक्स (Linux) सिस्टम बनाया गया था।
- सन् 2013 में एम्बेडेड बाजार $ 140 बिलियन तक पहुँच गया।
- विश्लेषकों का कहना है, कि वर्ष 2030 तक एम्बेडेड बाजार 40 बिलियन डॉलर से बड़ा हो जाएगा।

3.2 एम्बेडेड सिस्टम के प्रकार (Types of Embedded Systems)



एम्बेडेड सिस्टम को उनकी क्षमता और कार्यकारी आवश्यकताओं (Functional requirements) के आधार पर चार श्रेणियों में वर्गीकृत किया गया है, जो निम्न है—

1. स्टैंड एलोन एम्बेडेड सिस्टम (Stand Alone Embedded System)
2. रियल टाइम एम्बेडेड सिस्टम (Real Time Embedded System)
3. नेटवर्क्ड एम्बेडेड सिस्टम (Networked Embedded System)
4. मोबाइल एम्बेडेड सिस्टम (Mobile Embedded System)

3.2.1 स्टैंड एलोन एम्बेडेड सिस्टम (Stand Alone Embedded System)

स्टैंड अलोन एम्बेडेड सिस्टम को कम्प्यूटर की तरह होस्ट सिस्टम की आवश्यकता नहीं होती है यह अपना कार्य स्वयं करने में सक्षम होता है, यह इनपुट पोर्ट से एनालॉग या डिजिटल या इनपुट की प्रक्रिया लेता है, और उसके बाद डेटा की गणना और रूपांतरण करता है, और संयोजित डिवाइस के माध्यम से परिणामी डाटा देता है, जो या तो संयोजित डिवाइस को नियंत्रित या ड्राइव करता है, और फिर उसको प्रदर्शित करता है। MP प्लेयर, डिजिटल कैमरा, वीडियो गेम कंसोल, माइक्रोवेव ओवन आदि स्टैंड एलोन एम्बेडेड सिस्टम के उदहारण हैं।

3.2.2. रीयल टाइम एम्बेडेड सिस्टम (Real Time Embedded System)

एक रीयल-टाइम एम्बेडेड सिस्टम Strictly time specific है, जिसका अर्थ है कि यह एम्बेडेड सिस्टम एक विशेष/परिभाषित समय अंतराल में आउटपुट प्रदान करता हैं। इस प्रकार के एम्बेडेड सिस्टम महत्वपूर्ण परिस्थितियों में त्वरित प्रतिक्रिया प्रदान करते हैं, जो समय पर आधारित कार्य प्रदर्शन और उत्पादन की प्राथमिकता को सबसे अधिक प्राथमिकता देते हैं। यही कारण है, कि रक्षा क्षेत्र, चिकित्सा और स्वास्थ्य देखभाल क्षेत्र, और कुछ अन्य औद्योगिक अनुप्रयोगों में जहाँ रीयल समय में आउटपुट को अधिक महत्व दिया जाता है, वहाँ रीयल टाइम एम्बेडेड सिस्टम का उपयोग किया जाता है। ट्राफिक कण्ट्रोल सिस्टम, डिफेन्स सेक्टर और स्वास्थ्य सेक्टर में इनका उपयोग किया जाता है ।

इस रीयल-टाइम एम्बेडेड सिस्टम को दो प्रकारों में विभाजित किया गया है ।

1. सॉफ्ट रीयल टाइम एम्बेडेड सिस्टम (Soft Real Time Embedded System)—इस प्रकार के एम्बेडेड सिस्टम में समय/समय सीमा का कड़ाई से पालन नहीं किया जाता है। यदि कार्य की समय सीमा समाप्त हो गई है (इसका अर्थ है कि सिस्टम निर्धारित समय में परिणाम नहीं देता है) तब भी परिणाम या आउटपुट स्वीकार किया जाता है।

2. हार्ड रीयल-टाइम एम्बेडेड सिस्टम (Hard Real-Time Embedded System)—इस प्रकार के एम्बेडेड सिस्टम में कार्य के समय/समय सीमा का सख्ती से पालन किया जाता है। कार्य समय सीमा (निर्धारित समय अंतराल) के बीच पूरा किया जाना चाहिए अन्यथा परिणाम/आउटपुट स्वीकार नहीं किया जाता है।

3.2.3 नेटवर्क्ड एम्बेडेड सिस्टम (Networked Embedded System)

इस प्रकार के एम्बेडेड सिस्टम संसाधनों तक पहुँचने के लिए एक नेटवर्क से सम्बंधित होते हैं। यह संयोजित नेटवर्क LAN, WAN या Internet हो सकता है, कनेक्शन वायर्ड या वायरलेस किसी भी तरह से हो सकता है । इस प्रकार का एम्बेडेड सिस्टम का वर्तमान समय के एम्बेडेड सिस्टम अनुप्रयोगों में प्रयोग किया जाता है । एम्बेडेड वेब सर्वर एक प्रकार की प्रणाली है, जिसमें सभी एम्बेडेड डिवाइस एक वेब सर्वर से जुड़े होते हैं, और उनको वेब ब्राउज़र द्वारा एक्सेस और नियंत्रित किया जाता है और ये TCP/IP प्रोटोकॉल पर चलते हैं । होम सिक्योरिटी सिस्टम, ATM मशीन,कार्ड स्वाइप मशीन इसके उदहारण है ।

3.2.4 मोबाइल एम्बेडेड सिस्टम (Mobile Embedded System)

मोबाइल एम्बेडेड सिस्टम छोटे और प्रयोग करने में आसान हैं और इसके लिए कम संसाधनों की आवश्यकता होती है। ये सबसे पसंदीदा एम्बेडेड सिस्टम हैं। पोर्टेबिलिटी के दृष्टिकोण में मोबाइल एम्बेडेड सिस्टम भी सबसे अच्छा है मोबाइल एम्बेडेड सिस्टम का उपयोग Portable embedded devices जैसे सेल फोन, मोबाइल, डिजिटल कैमरा, एमपी 3 प्लेयर और पर्सनल डिजिटल असिस्टेंट आदि में किया जाता है ।

प्रदर्शन (Performance) और माइक्रो-कंट्रोलर के आधार पर इसे तीन प्रकार में विभाजित किया गया है—

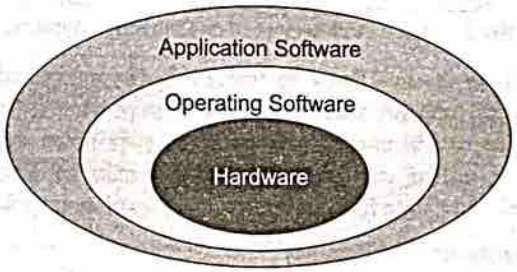
1. छोटे स्तर के एम्बेडेड सिस्टम (Small Scale Embedded System)—स्केल एम्बेडेड सिस्टम 8-बिट या 16-बिट माइक्रो-कंट्रोलर का उपयोग करके बनाया गया है। इन्हें बैटरी द्वारा संचालित किया जा सकता है। यह प्रोसेसर मेमोरी और प्रोसेसिंग स्पीड के बहुत कम/सीमित संसाधनों का उपयोग करता है। मुख्य रूप से सिस्टम एक स्वतंत्र प्रणाली के रूप में कार्य नहीं करते हैं, ये कम्प्यूटर सिस्टम के किसी भी घटक के रूप में कार्य करते हैं, लेकिन उन्होंने एक विशिष्ट कार्य के लिए समर्पित किया जा सकता है पर ये गणना नहीं करते है।

2. मध्यम स्तर के एम्बेडेड सिस्टम (Medium Scale Embedded Systems)—मध्यम स्तर के एम्बेडेड सिस्टम 16-बिट या 32-बिट माइक्रो-कंट्रोलर का उपयोग करके बनाए गए हैं। ये मध्यम स्तर एम्बेडेड सिस्टम छोटे स्तर के एम्बेडेड सिस्टम की तुलना में तेज़ होते हैं। इन प्रणालियों में हार्डवेयर और सॉफ्टवेयर का एकीकरण जटिल होता है। JAVA, C, व C++ ऐसी प्रोग्रामिंग भाषाएँ हैं, जिनका प्रयोग मध्यम स्तर के एम्बेडेड सिस्टम को विकसित करने के लिए किया जाता है। इस प्रकार के सिस्टम को विकसित करने के लिए विभिन्न प्रकार के सॉफ्टवेयर टूल; जैसे—कंपाइलर, डीबगर, सिम्युलेटर आदि का उपयोग किया जाता है।

3. कॉम्प्लैक्स एम्बेडेड सिस्टम (Sophisticated or Complex Embedded Systems)—परिष्कृत या जटिल एम्बेडेड सिस्टम 32-बिट या 64-बिट माइक्रो-नियंत्रक का उपयोग करके बनाए किए गए हैं। इन प्रणालियों को बड़े पैमाने पर जटिल कार्यों को करने के लिए विकसित किया जाता है। इन प्रणालियों में उच्च हार्डवेयर और सॉफ्टवेयर जटिलताएँ होती हैं। हम अंतिम सिस्टम या हार्डवेयर उत्पादों को डिजाइन करने के लिए हार्डवेयर और सॉफ्टवेयर दोनों घटकों का उपयोग करते हैं।

3.3 एम्बेडेड सिस्टम आर्किटेकचर (Embedded Systems Architecture)

एम्बेडेड दृष्टिकोण कुछ है, जो किसी अन्य पहलू से जुड़ा हुआ है। एक एम्बेडेड गैजेट एक लैपटॉप हार्डवेयर डिवाइस के रूप में धारणा हो सकती है, जिसमें सॉफ्टवेयर प्रोग्राम एम्बेडेड है। एक एम्बेडेड गैजेट एक निष्पक्ष सिस्टम हो सकता है या यह एक विशाल सिस्टम का हिस्सा हो सकता है। एक एम्बेडेड प्रणाली एक माइक्रोकंट्रोलर या माइक्रोप्रोसेसर मुख्य रूप से आधारित गैजेट है, जो किसी विशेष कार्य को करने के लिए बनाया गया है। उदाहरण के लिए, फायर अलार्म एक एम्बेडेड सिस्टम है।



चित्र 3.3 : Architecture of embedded system

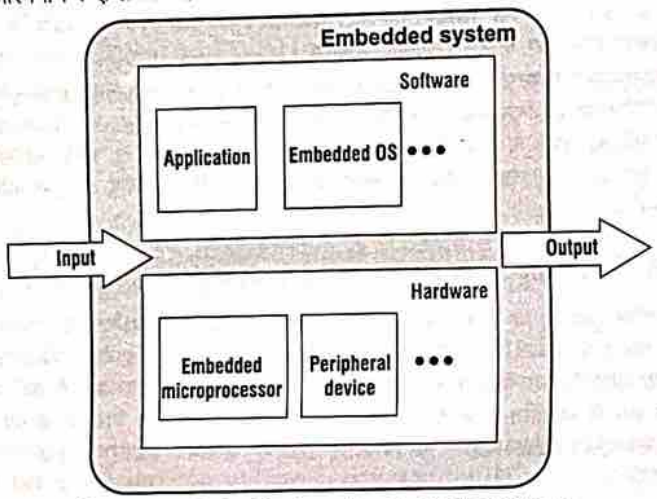
एक एम्बेडेड सिस्टम के आर्किटेकचर में तीन घटक हार्डवेयर, सॉफ्टवेयर और इसमें एक वास्तविक रीयल-टाइम ऑपरेटिंग सिस्टम (RTOS) होता है, जो यूटिलिटी सॉफ्टवेयर की देख-रेख करता है, और प्रोसीजर को लेट करने के लिए एक प्लान को फॉलो करने के माध्यम से प्रोसेसर को प्रोसेस के रूप में प्रोसेस करने देने के लिए एक तंत्र देता है। RTOS सिस्टम के कार्य करने के तरीके को परिभाषित करता है। यह एप्लिकेशन सॉफ़्टवेयर के निष्पादन के दौरान नियमों को पूरा करता है। एक छोटे स्तर के एम्बेडेड डिवाइस में RTOS नहीं होता है

इसलिए हम एक एम्बेडेड गैजेट को एक माइक्रोकंट्रोलर के रूप में पूरी तरह से सॉफ्टवेयर, भरोसेमंद, वास्तविक समय प्रबंधित डिवाइस के रूप में परिभाषित करेंगे।

नीचे दी गई तालिका में कुछ सामान्य सिस्टम का सार दिया गया है, जो एम्बेडेड आर्किटेक्चर का निर्माण कर सकती हैं, और आमतौर पर यह इंगित करती हैं, कि एक विशिष्ट संरचना के तत्व क्या दर्शाते हैं और ये कारक कैसे

परस्पर संबंध रखते हैं। यह अतिरिक्त रूप से एक अंतर्निहित प्रणाली का प्रतीक होने के लिए वास्तु संरचनाओं की विशाल शैली को दर्शाता है। आर्किटेकचर और उनकी संरचनाएँ, वे कैसे परस्पर संबंध रखते हैं, आर्किटेकचर कैसे बनाते हैं।

चित्र दो मुख्य भागों में युक्त एक विशिष्ट एम्बेडेड सिस्टम का कॉन्फिगरेशन आरेख दिखाता है : एम्बेडेड हार्डवेयर और एम्बेडेड सॉफ्टवेयर। एम्बेडेड हार्डवेयर में मुख्य रूप से प्रोसेसर, मेमोरी, बस, परिधीय उपकरण (peripheral devices), I/O पोर्ट और विभिन्न नियंत्रक शामिल हैं। एम्बेडेड सॉफ्टवेयर में आमतौर पर एम्बेडेड ऑपरेटिंग सिस्टम और विभिन्न एप्लिकेशन होते हैं।



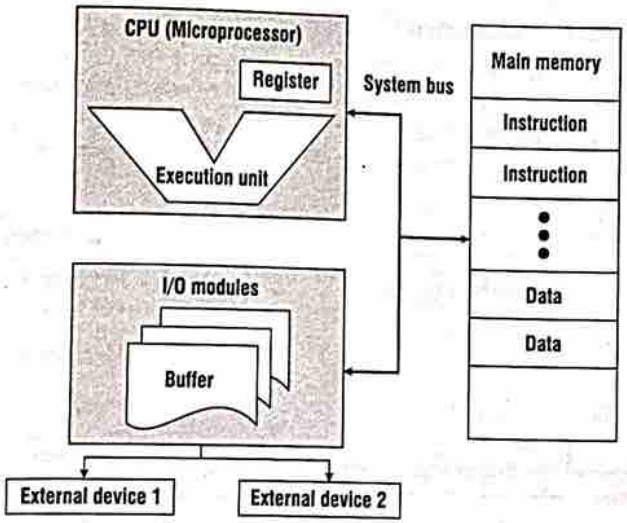
चित्र 3.4 : Basic Architecture of an Embedded System

इनपुट और आउटपुट किसी भी ओपन सिस्टम की विशेषताएँ हैं, और एम्बेडेड सिस्टम कोई अपवाद नहीं है। एम्बेडेड सिस्टम में, हार्डवेयर और सॉफ्टवेयर अक्सर बाहर से विभिन्न इनपुट संकेतों से निपटने के लिए सहयोग करते हैं और प्रोसेसिंग फॉर्म को कुछ फॉर्म के माध्यम से आउटपुट देते हैं। इनपुट सिग्नल एक एर्गोनोमिक डिवाइस, जैसे की-बोर्ड, माउस या टच स्क्रीन, या किसी अन्य एम्बेडेड सिस्टम में सेंसर सर्किट का आउटपुट हो सकते हैं। आउटपुट ध्वनि, प्रकाश, बिजली या किसी अन्य एनालॉग सिग्नल या डेटाबेस के लिए रिकॉर्ड या फाइल के रूप में हो सकता है।

हार्डवेयर आर्किटेकचर (Hardware Architecture)

बुनियादी कम्प्यूटर प्रणाली के घटक-माइक्रोप्रोसेसर, मेमोरी और इनपुट और आउटपुट मॉड्यूल होते हैं ये सभी एक सिस्टम बस द्वारा सभी भागों को एक प्रोग्राम को संप्रेषित करने और निष्पादित करने के लिए परस्पर जुड़े हुए होते हैं।

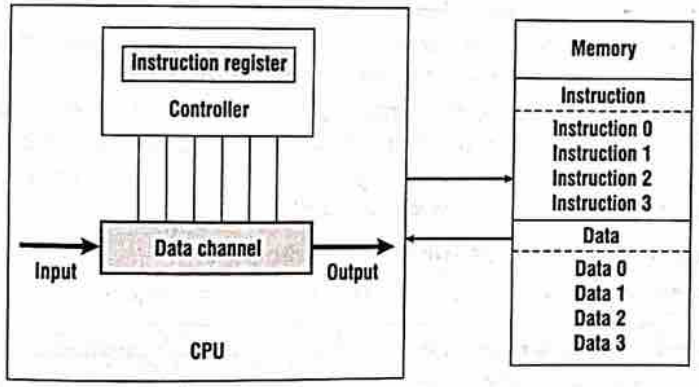
एम्बेडेड सिस्टम में, माइक्रोप्रोसेसर की भूमिका और फंक्शन आमतौर पर general-purpose कंप्यूटर में सीपीयू के समान होते हैं और कंप्यूटर ऑपरेशन को नियंत्रित करते हैं, निर्देशों को निष्पादित करते हैं और डेटा को प्रोसेस करते हैं। कई मामलों में, एक एम्बेडेड सिस्टम में माइक्रोप्रोसेसर को सीपीयू भी कहा जाता है। मेमोरी का उपयोग निर्देशों और डेटा को स्टोर करने के लिए किया जाता है। I/O मॉड्यूल प्रोसेसर, मेमोरी और बाहरी उपकरणों के बीच डेटा विनिमय के लिए जिम्मेदार होते हैं। बाहरी उपकरणों में सेकंडरी मेमोरी उपकरण (जैसे फ्लैश और हार्ड डिस्क), संचार उपकरण और टर्मिनल उपकरण सम्मिलित हैं। सिस्टम बस डेटा प्रदान करता है और प्रोसेसर, मेमोरी और I/O मॉड्यूल के लिए सिग्नल संचार और ट्रांसमिशन को नियंत्रित करता है।



चित्र 3.5 : Computer Architecture

वॉन न्यूमैन आर्किटेकचर (Von Neumann Architecture)

वॉन न्यूमैन आर्किटेकचर (प्रिंसटन आर्किटेकचर के रूप में भी जाना जाता है) पहली बार जॉन वॉन न्यूमैन द्वारा प्रस्तावित किया गया था। इस आर्किटेकचर की सबसे बड़ी खासियत यह है कि सॉफ्टवेयर और डेटा एक-ही मेमोरी का उपयोग करते हैं; अर्थात् 'प्रोग्राम डेटा है, और डेटा प्रोग्राम है'।

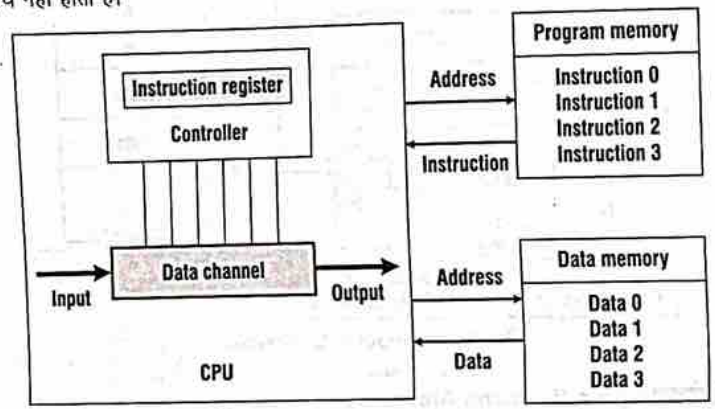


चित्र 3.6 : Von Neumann Architecture

वॉन न्यूमैन आर्किटेकचर में, एक निर्देश और डेटा एक ही बस को साझा करते हैं। इस आर्किटेकचर में, सूचना का प्रसारण कम्प्यूटर के डिस्प्ले की hindrance बन जाता है और डेटा प्रोसेसिंग की गति को प्रभावित करता है: इसलिए, इसे अक्सर वॉन न्यूमन बोतलनैक भी कहा जाता है। वास्तव में, cache और ब्रांच-प्रोडक्शन तकनीक इस मुद्दे को प्रभावी ढंग से हल कर सकते हैं।

हार्वर्ड आर्किटेकचर (Harvard Architecture)

हार्वर्ड आर्किटेकचर का नाम पहली बार हार्वर्ड मार्क-I कम्प्यूटर के नाम पर रखा गया था। वॉन न्यूमैन आर्किटेकचर की तुलना में, हार्वर्ड आर्किटेकचर प्रोसेसर में दो उत्कृष्ट विशेषताएँ हैं। सबसे पहले, निर्देश और डेटा दो अलग-अलग मेमोरी मॉड्यूल में संग्रहीत किए जाते हैं; निर्देश और डेटा एक-ही माड्यूल में सह-अस्तित्व नहीं रखते हैं। दूसरा, दो स्वतंत्र बसों को सीपीयू और मेमोरी के बीच समर्पित संचार पथ के रूप में उपयोग किया जाता है; दोनों बसों के बीच कोई संबंध नहीं होता है।



चित्र 3.7 : Harvard Architecture

क्योंकि हार्वर्ड आर्किटेकचर के पास अलग-अलग प्रोग्राम मेमोरी और डैटा मेमोरी हैं, यह अधिक डेटा-मेमोरी बैंडविड्थ प्रदान कर सकता है, जिससे यह डिजिटल सिग्नल प्रोसेसिंग के लिए आदर्श विकल्प बन सकता है। डिजिटल सिग्नल प्रोसेसिंग (डीएसपी) के लिए डिजाइन किए गए अधिकांश सिस्टम हार्वर्ड आर्किटेकचर को अपनाते हैं। वॉन न्यूमैन आर्किटेकचर में सरल हार्डवेयर डिजाइन और लचीले प्रोग्राम और डेटा स्टोरेज की सुविधा है और इसे आमतौर पर सामान्य उद्देश्य और सबसे एम्बेडेड सिस्टम के लिए चुना जाता है।

मेमोरी में कुशलतापूर्वक पढ़ने/लिखने के लिए, प्रोसेसर मुख्य मेमोरी से सीधे जुड़ा नहीं होता है, लेकिन cache से डायरेक्ट जुड़ा होता है। आमतौर पर, हार्वर्ड आर्किटेकचर और वॉन न्यूमैन आर्किटेकचर के बीच एकमात्र अंतर एकल या दोहरी L1 cache है। हार्वर्ड आर्किटेकचर में, L1 cache अक्सर एक निर्देश cache (I cache) और डेटा कैश (D cache) में विभाजित होता है, लेकिन वॉन न्यूमैन आर्किटेकचर में एक-ही cache होता है।

Von-Neumann Architecture vs Harvard Architecture

निम्नलिखित बिंदु हार्वर्ड आर्किटेकचर से वॉन न्यूमैन आर्किटेकचर को अलग करते हैं।

Von-Neumann Architecture	Harvard Architecture
इसमें कोड और डेटा दोनों द्वारा साझा की जाने वाली एकल मेमोरी होती है।	इसमें कोड और डेटा दोनों द्वारा साझा की जाने वाली मेमोरी अलग-अलग होती है।
प्रोसेसर को एक क्लॉक साइकिल में कोड और दूसरे क्लॉक साइकिल में डेटा प्राप्त करने की आवश्यकता होती है तब इसके लिए दो क्लॉक साइकिल की आवश्यकता होती है।	इसमें कोड व डाटा को एक्सेस करने के लिए एक क्लॉक साइकिल ही काफी होती है।
इनकी उच्च गति होती है, इसलिए इसमें कम समय लगता है।	इनमें गति कम होती है इसलिए बहुत समय लगता है।
ये डिजाइन में सरल होते हैं।	ये डिजाइन में जटिल होते हैं।

एम्बेडेड सिस्टम का माइक्रोप्रोसेसर आर्किटेकचर (Microprocessor Architecture of Embedded Systems)

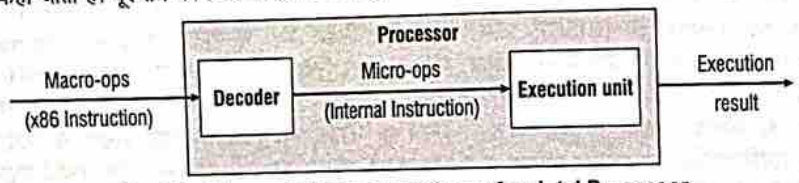
एम्बेडेड सिस्ट में माइक्रोप्रोसेसर मुख्य भाग होता है। एक विशेष सर्किट बोर्ड में माइक्रोप्रोसेसर स्थापित करके और आवश्यक परिधीय सर्किट और विस्तार सर्किट को जोड़कर, एक व्यावहारिक एम्बेडेड सिस्टम बनाया जा सकता है। माइक्रोप्रोसेसर आर्किटेकचर निर्देश का निर्धारण करता है, परिधीय सर्किट और विस्तार सर्किट का समर्थन करता है। माइक्रोप्रोसेसरों की 4-, 8-, 16-, 32- और 64- बिट, मेगाहट्र्ज से गीगाहट्र्ज तक के डिस्प्ले के साथ, और कुछ पिन से लेकर हजारों पिन तक एक विस्तृत श्रृंखला है।

सामान्यतौर पर, दो प्रकार के एम्बेडेड माइक्रोप्रोसेसर आर्किटेकचर होते हैं: कम निर्देश सेट कम्प्यूटर {Reduced Instruction Set Computer (RISC)} और कॉम्पलैक्स निर्देश सेट कम्प्यूटर {Complex Instruction Set Computer (CISC)}। RISC प्रोसेसर एक छोटा, सीमित, सरल निर्देश सेट का उपयोग करता है। प्रत्येक निर्देश एक मानक शब्द लम्बाई का उपयोग करता है और निष्पादन समय छोटा होता है, जो निर्देश पाइपलाइन के अनुकूलन की सुविधा देता है। कमांड फंक्शन के लिए क्षतिपूर्ति करने के लिए, सीपीयू अक्सर बड़ी संख्या में सामान्य उद्देश्य रजिस्टरों से सुसज्जित होता है। CISC प्रोसेसर एक शक्तिशाली अनुदेश सेट और विभिन्न निर्देश लम्बाई प्रदान करता है, जो निर्देशों के विखंडित निष्पादन की सुविधा देता है। RISC और CISC की तुलना तालिका में दी गई है।

Table : Comparison of RISC and CISE

	RISC	CISC
Instruction system	Simple and efficient instructions. Realizes uncommon functions through combined instructions.	Rich instruction system. Performs specific functions through special instructions; handles special tasks efficiently.
Memory operation	Restricts the memory operation and simplifies the controlling function.	Has multiple memory operation instructions and performs direct operation.
Program	Requires a large amount of memory space for the assembler and features complex programs for special functions.	Has a relatively simple assembler and features easy and efficient programming of scientific computing and complex operations.
Interruption	Responds to an interrupt only at the proper place in instruction execution.	Responds to an interruption only at the end of execution.
CPU	Features fewer unit circuits, small size, and low power consumption.	Has feature-rich circuit units, powerful functions, a large area and high power consumption.
Design cycle	Features a simple structure, a compact layout, a short design cycle and easy application of new technologies.	Features a complex structure and long design cycle.
Usage	Features a simple structure, regular instructions, simple control and easy learning and application.	Features a complex structure, powerful functions and easy realization of special functions.
Application scope	Determines the instruction system per specific areas, which is more suitable for special machines.	Becomes more suitable for general-purpose machines.

RISC और CISC की अलग-अलग विशेषताएँ और लाभ हैं, लेकिन RISC और CISC के बीच की सीमाएँ माइक्रोप्रोसेसर क्षेत्र में धुंधली पढ़ने लगती हैं। कई पारंपरिक CISCs RISC के लाभों को अवशोषित करते हैं और RISC जैसी डिजाइन का उपयोग करते हैं। Intel x86 प्रोसेसर उनमें से विशिष्ट हैं। उन्हें CISC आर्किटेकचर माना जाता है। ये प्रोसेसर एक डिकोडर के माध्यम से RISC जैसे निर्देशों में x86 निर्देशों का अनुवाद करते हैं और RISC आर्किटेकचर के लाभों को प्राप्त करने और आंतरिक संचालन दक्षता में सुधार करने के लिए RISC डिजाइन और संचालन का अनुपालन करते हैं। एक प्रोसेसर के आंतरिक अनुदेश निष्पादन को माइक्रो-ऑपरेशन कहा जाता है, जिसे माइक्रो-ओपी और संक्षिप्त म्यू-ऑप (m-op या mop) के रूप में दर्शाया जाता है। इसके विपरीत, x86 निर्देश को मैक्रो-ऑप कहा जाता है। पूरे तंत्र को चित्र में दर्शाया गया है।



चित्र 3.8 : Micro and Macro Operations of an Intel Processor

आमतौर पर, एक मैक्रो ऑपरेशन को निष्पादित करने के लिए एक या एक-से अधिक माइक्रो ऑपरेशन में डिकोड किया जा सकता है, लेकिन कभी-कभी एक डिकोडर निष्पादित करने के लिए एक माइक्रो ऑपरेशन उत्पन्न करने के लिए, कई मैक्रो ऑपरेशन को जोड़ सकता है। इस प्रक्रिया को x86 निर्देश फ्यूजन (मैक्रो-ऑप्स फ्यूजन) के रूप में जाना जाता है। उदाहरण के लिए, प्रोसेसर x86 CMP (comparision) निर्देश और x86 JMP (Jump) निर्देश को एक-ही माइक्रो ऑपरेशन, comparision और jump के निर्देश को जोड़ सकता है। इस संयोजन के स्पष्ट लाभ हैं कम निर्देश, जो अप्रत्यक्ष रूप से प्रोसेसर निष्पादन के प्रदर्शन को बढ़ाता है और फ्यूजन प्रोसेसर को निर्देशों के बीच समानता को अधिकतम करने में सक्षम बनाता है और परिणामस्वरूप प्रोसेसर की कार्यान्वयन दक्षता में सुधार करता है।

वर्तमान में, अधिकांश एम्बेडेड सिस्टम में उपयोग किए जाने वाले माइक्रोप्रोसेसर में पाँच आर्किटेकचर हैं—

RISC, CISC, MIPS, PowerPC और SuperH जिनका विवरण निम्न प्रकार है—

MIPS Acchitecture : Microprocessor without Interlocked Piped Stages (MIPS) भी एक RISC प्रोसेसर है। इसका तंत्र पाइपलाइन में डेटा issues से बचने के लिए सॉफ्टवेयर का पूर्ण उपयोग करना है। यह पहली बार 1980 के दशक की शुरुआत में स्टेनफोर्ड विश्वविद्यालय के प्रोफेसर जॉन हेनेसी के नेतृत्व में एक शोध दल द्वारा विकसित किया गया था और बाद में इसका संचालन NIPS टेक्नोलॉजीस द्वारा किया गया था। ARM की तरह, MIPS टेक्नोलॉजी, आईपी (IP) कोर के माध्यम से अर्धचालक कम्पनियों को MIPS माइक्रोप्रोसेसर कोर प्रदान करता है और उन्हें RISC आर्किटेकचर में एम्बेडेड माइक्रोप्रोसेसर विकसित करने की अनुमति देता है। प्रोसेसर में waste processing units को विभाजित करके दूसरे कोर के रूप में वर्चुअलाइज करने और processing units के उपयोग में सुधार करने के लिए Core technology एक multi-issue capability है।

PowerPC Architecture : PowerPC RISC आर्किटेकचर में एक सीपीयू है। यह पॉवर आर्किटेकचर से व्युत्पन्न है, और इसका मूल डिजाइन IBM PowerPC 601 माइक्रोप्रोसेसर डिस्प्ले से आया है जो एन्हांस्ड RISC (Power) के साथ अनुकूलित है। 1990 के दशक में, IBM, Apple और Motorola ने सफलतापूर्वक PowerPC चिप विकसित की और एक PowerPC आधारित मल्टीप्रोसेसर कम्प्यूटर बनाया। PowerPC आर्किटेकचर में स्केलेबिलिटी, सुविधा, flexibility और Openness है, यह एक निर्देश सेट आर्किटेकचर (आईएसए) को परिभाषित करता है, जो किसी को भी PowerPC-संगत प्रोसेसर डिजाइन और निर्माण करने की अनुमति देता है, और PowerPC के लिए विकसित सॉफ्टवेयर मॉड्यूल के स्रोत कोड का स्वतंत्र रूप से उपयोग करता है। PowerPC में संचार और नेटवर्किंग क्षेत्र जैसे, स्विच, राउटर इत्यादि के साथ मोबाइल फोन से गेम कंसोल तक की एक विस्तृत श्रृंखला है।

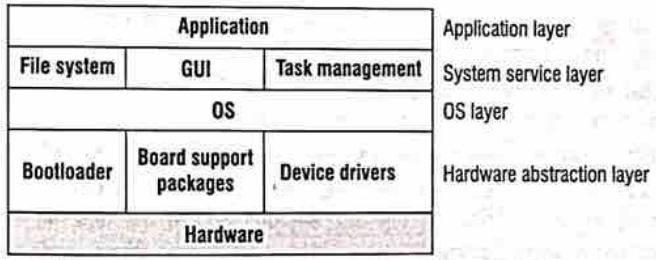
Apple Mac Series ने एक दशक तक PowerPC प्रोसेसर का उपयोग किया जब तक कि Apple ने x86 आर्किटेकचर पर स्विच नहीं किया।

SuperH : SuperH (SH) एक अत्यधिक लागत प्रभावी (cost affecting), कॉम्पैक्ट, एम्बेडेड RISC प्रोसेसर है। SH आर्किटेकचर पहली बार Hitachi द्वारा विकसित किया गया था और Hitachi और ST माइक्रोइलेक्ट्रॉनिक के स्वामित्व में था। अब इसे रेनेसर ने संभाल लिया है। SuperH में SH-1, SH-2, SH-DSP, SH-3, SH-3-DSP, SH-4, SH-5 और SH-X श्रृंखला शामिल है और व्यापक रूप से प्रिंटर, फैक्स, मल्टीमीडिया टर्मिनल, टीवी गेम में, कंसोल, सेट-टॉप बॉक्स, सीडी-रोम, घरेलू उपकरण और अन्य एम्बेडेड सिस्टम में उपयोग किया जाता है।

सॉफ्टवेयर आर्किटेकचर (Software Architecture)

एम्बेडेड हार्डवेयर की तरह, एम्बेडेड सॉफ्टवेयर आर्किटेकचर अत्यधिक flexible है। सरल एम्बेडेड सॉफ्टवेयर (जैसे इलेक्ट्रॉनिक खिलौने, कैलकुलेटर इत्यादि) कोड की केवल कुछ हजार लाइनें हो सकती हैं और सरल इनपुट और आउटपुट फंक्शन कर सकती हैं। दूसरी ओर, जटिल एम्बेडेड सिस्टम (जैसे स्मार्टफोन, रोबोट इत्यादि) को डेस्कटॉप कम्प्यूटर और सर्वर के समान अधिक जटिल सॉफ्टवेयर आर्किटेकचर की आवश्यकता होती है। सरल एम्बेडेड सॉफ्टवेयर low display chip हार्डवेयर के लिए उपयुक्त है, इसमें बहुत limited functionality है, और tedious secondary development की आवश्यकता होती है। जटिल एम्बेडेड सिस्टम अधिक शक्तिशाली कार्य प्रदान करते हैं, उपयोगकर्ताओं के लिए अधिक सुविधाजनक इंटरफेस की आवयश्यकता होती है और more powerful हार्डवेयर support की आवश्यकता होती है। हार्डवेयर integration और processing क्षमताओं में सुधार के साथ, हार्डवेयर bottlenecks gradually relaxed और even broken, इसलिए एम्बेडेड सिस्टम सॉफ्टवेयर अब पूरी तरह कार्यात्मक और विविध हो जाता है।

पूर्ण एम्बेडेड सिस्टम सॉफ्टवेयर को चित्र में दिखाया गया है—



चित्र 3.9 : Software Architecture of an Embedded System

एक एम्बेडेड सॉफ्टवेयर सिस्टम नीचे से ऊपर तक चार layers में बना होता है—

1. हार्डवेयर एब्सट्रैक्शन लेयर (Hardware Abstraction Layer)
2. ऑपरेटिंग सिस्टम लेयर (Operating System Layer)
3. सिस्टम सर्विस लेयर (System Service Layer)
4. एप्लीकेशन लेयर (Application Layer)

1. हार्डवेयर एब्स्ट्रैक्शन लेयर (Hardware Abstraction Layer)—OS के एक भाग के रूप में हार्डवेयर एब्स्ट्रैक्शन लेयर (HAL), एम्बेडेड सिस्टम हार्डवेयर और OS के बीच सॉफ्टवेयर एब्स्ट्रैक्शन लेयर होती है। सामान्य तौर पर, HAL में बूटलोडर, बोर्ड सपोर्ट पैकेज (BSP), डिवाइस ड्राइवर और अन्य घटक शामिल होते हैं। पीसी में BIOS के समान, बूटलोडर एक प्रोग्राम है जो OS कर्नेल निष्पादित होने से पहले चलता है। यह हार्डवेयर के आरंभीकरण को पूरा करता है, मेमोरी स्पेस की इमेज स्थापित करता है और परिणामस्वरूप हार्डवेयर और सॉफ्टवेयर वातावरण को सिस्टम कर्नेल के अंतिम निर्धारण के लिए एक उपयुक्त स्थिति तक पहुँचने में सक्षम बनाता है। अंतिम

उपयोगकर्त्ताओं के दृष्टिकोण से, बूटलोडर का उपयोग OS को लोड करने के लिए किया जाता है। बीएसपी हार्डवेयर ऑपरेशन के एब्स्ट्रैक्ट होने को प्राप्त करता है, OS को हार्डवेयर से स्वतंत्र होने और OS को विभिन्न हार्डवेयर आर्किटेकचर पर चलने के लिए सक्षम बनाता है। प्रत्येक OS के लिए एक unique BSP बनाया जाना चाहिए। उदाहरण के लिए, विंड रिवर VxWorks BSP और माइक्रोसॉफ्ट विंडोज सीई बीएसपी में एक एम्बेडेड हार्डवेयर डेवलपमेंट बोर्ड के समान कार्य है, लेकिन वे पूरी तरह से अलग आर्किटेकचर और इंटरफेस की सुविधा देते हैं। बीएसपी की अवधारणा का उल्लेख शायद ही कभी किया जाता है जब विभिन्न डेस्कटॉप विंडोज या Linux OS पर चर्चा की जाती है, क्योंकि सभी PC इंटीग्रेटिड इंटेल आर्किटेकचर को अपनाते हैं: OS को आसानी से बिना किसी बदलाव के विविध इंटेल आर्किटेकचर आधारित उपकरणों में माइग्रेट किया जा सकता है। एम्बेडेड सिस्टम में BSP एक unique सॉफ्टवेयर माड्यूल है। इसके अलावा, डिवाइस ड्राइवर OS को हार्डवेयर घटकों और बाह्य उपकरणों के बीच अंतर को ढालने में सक्षम करते हैं और ऑपरेटिंग हार्डवेयर के लिए integrated software interface प्रदान करते हैं।

2. ऑपरेटिंग सिस्टम लेयर (Operating System Layer)—OS एक समान रूप से हार्डवेयर संसाधनों के प्रबंधन के लिए एक सॉफ्टवेयर सिस्टम है। यह कई हार्डवेयर फंक्शन को एब्सट्रैक्ट करता है और उन्हें सर्विस के रूप में एप्लिकेशन प्रदान करता है। Determination, Files synchronization और Networking OS द्वारा प्रदान की जाने वाली सबसे ordinary services हैं। ऑपरेटिंग सिस्टम व्यापक रूप से अधिकांश डेस्कटॉप और एम्बेडेड सिस्टम में उपयोग किया जाता है। एम्बेडेड सिस्टम में, Stability, Customization, Modularity और Real Time Processing OS की अपनी अनूठी विशेषताएँ हैं। सामान्य एम्बेडेड OS में एम्बेडेड Linux, Windows, CE, VxWorks, MeeGo, Tizen, Android, Ubuntu और कुछ ऑपरेटिंग सिस्टम विशिष्ट क्षेत्रों में उपयोग किए जाते हैं। एम्बेडेड Linux कर्नेल है जिसे मोबाइल और एम्बेडेड उत्पादों के लिए अनुकूलित और संशोधित किया गया है। विंडोज सीई एक अनुकूलन योग्य एम्बेडेड OS है जिसे माइक्रोसॉफ्ट ने कई एम्बेडेड सिस्टम और उत्पादों के लिए लॉन्च किया है। VxWorks, विंड रिवर से एक एम्बेडेड रियल टाइम ऑपरेटिंग सिस्टम (RTOS), PowerPC, 68K, CPU32, SPARC, 1960, x86, ARM और MIPS का समर्थन करता है। Outstanding real time और विश्वसनीय सुविधाओं के साथ, यह व्यापक रूप से संचार, सैन्य, एयरोस्पेस, विमानन और अन्य क्षेत्रों में उपयोग किया जाता है जिनके लिए Highly sophisticated, Real time technologies की आवश्यकता होती है। विशेष रूप से, नासा द्वारा मार्स प्रोब में VWWorks का उपयोग किया जाता है।

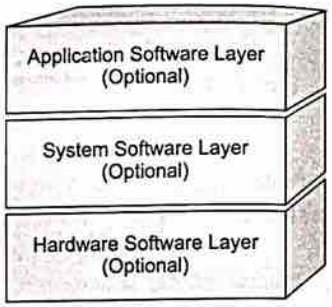
3. सिस्टम सर्विस लेयर (System Service Layer)—सिस्टम सर्विस लेयर वह सर्विस इंटरफेस है जो OS applications को प्रदान करता है। इस इंटरफेस का उपयोग करते हुए, एप्लिकेशन OS द्वारा प्रदान की गई विभिन्न सर्विस तक पहुँच सकते हैं। कुछ हद तक, यह OS और अनुप्रयोगों के बीच एक लिंक की भूमिका निभाता है। इस लेयर में आमतौर पर फाइल सिस्टम, ग्राफिकल यूजर इंटरफेस (GUI), टास्क मैनेजर इत्यादि शामिल होते हैं। GUI लाइब्रेरी विभिन्न GUI प्रोग्रामिंग इंटरफेस के साथ एप्लिकेशन प्रदान करता है, जो एप्लिकेशन को कमांड लाइन के स्थान पर एप्लिकेशन विंडो, मेनू, डायलॉग बॉक्स और अन्य ग्राफिक रूपों के माध्यम से उपयोगकर्त्ताओं के साथ interact करने में सक्षम बनाता है।

4. एप्लीकेशन लेयर (Application layer)—सॉफ्टवेयर hierarchy के शीर्ष स्तर पर स्थित applications, सिस्टम कार्यक्षमता और व्यावसायिक तर्क को लागू करता है। एक functional दृष्टिकोण से, application में मॉड्यूल के सभी levels का purpose सिस्टम promote करना है। सिस्टम के दृष्टिकोण से, प्रत्येक एप्लिकेशन एक अलग OS प्रक्रिया है। आमतौर पर, एप्लिकेशन less privileged वाले प्रोसेसर मोड में चलते हैं और OS के साथ interact करने के लिए OS द्वारा प्रदान की गईं API प्रणाली अनुसूची का उपयोग करते हैं।

Structure Types*			Definition
Module			Elements (referred to as modules) are defined as the different functional components (the essential hardware and/or software that the system needs to function correctly) within an embedded device. Marketing and sales architectural diagrams are typically represented as modular structures, since software or hardware is typically packaged for sale as modules (*i.e.*, an operating system, a processor, a JVM, and so on).
	Uses (also referred to as subsystem and component)		A type of modular structure representing system at runtime in which modules are inter-related by their usages (what module uses what other module, for example).
		Layers	A type of Uses structure in which modules are organized in layers (*i.e.*, hierarchical) in which modules in higher layers use (require) modules of lower layers.
		Kernel	Structure presents modules that use modules (services) of an operating system kernel or are manipulated by the kernel.
		Channel Architecture	Structure presents modules sequentially, showing the module transformations through their usages.
		Virtual Machine	Structure presents modules that use modules of a virtual machine.
	Decomposition		A type of modular structure in which some modules are actually subunits (decomposed units) of other modules, and inter-relations are indicated as such. Typically used to determine resource allocation, project management (planning), data management (encapsulation, privitization, etc.).
	Class (also referred to as generalization)		This is a type of modular structure representing software and in which modules are referred to as classes, and inter-relationships are de?ned according to the object-oriented approach in which classes are inheriting from other classes, or are actual instances of a parent class (for example). Useful in designing systems with similar foundations.
Component and Connector			These structures are composed of elements that are either components (main hw/sw processing units, such as processors, a Java Virtual Machine, etc.) or connectors (communication mechanism that inter-connects components, such as a hw bus, or sw OS messages, etc.).
	Client/Server (also referred to as distribution)		Structure of system at runtime where components are clients or servers (or objects), and connectors are the mechanisms used (protocols, messages, packets, etc.) used to intercommunicate between clients and servers (or objects).
	Process (also referred to as communicating processes)		This structure is a SW structure of a system containing an operating system. Components are processes and/or threads (see Chapter 9 on OSes), and their connecters are the inter-process communication mechanisms (shared data, pipes, etc.) Useful for analyzing scheduling and performance.
	Concurrency and Resource		This structure is a runtime snap shot of a system containing an OS, and in which components are connected via threads running in parallel (see Chapter 9, Operating Systems). Essentially, this structure is used for resource management and to determine if there are any problems with shared resources, as well as to determine what sw can be executed in parallel.
		Interrupt	Structure represents the interrupt handling mechanisms in system.
		Scheduling (EDF, priority, round-robin)	Structure represents the task scheduling mechanism of threads demonstrating the fairness of the OS scheduler.
	Memory		This runtime representation is of memory and data components with the memory allocation and deallocation (connector) schemes—essentially the memory management scheme of the system.
		Garbage Collection	This structure represents the garbage allocation scheme (more in Chapter 2).
		Allocation	This structure represents the memory allocation scheme of the system (static or dynamic, size, and so on).
	Safety and Reliability		This structure is of the system at runtime in which redundant components (hw and sw elements) and their intercommunication mechanisms demonstrate the reliability and safety of a system in the event of problems (its ability to recover from a variety of problems).
Allocation			A structure representing relationships between sw and/or hw elements, and external elements in various environments.
	Work Assignment		This structure assigns module responsibility to various development and design teams. Typically used in project management.
	Implementation		This is a sw structure indicating where the sw is located on the development system's ?le system.
	Deployment		This structure is of the system at runtime where elements in this structure are hw and sw, and the relationship between elements are where the sw maps to in the hardware

3.4 एम्बेडेड सिस्टम का प्रारूप (Model of Embedded System)

एम्बेडेड सिस्टम आर्किटेक्चर संरचनाओं का विस्तार तकनीकी अवधारणाओं और एक एम्बेडेड डिवाइस के मूल सिद्धांतों को प्रस्तुत करने के लिए उपयोग किया जाता है। उभरते वास्तु उपकरण (Emerging architectural equipment) (यानी, संदर्भ मॉडल) का उपयोग इन वास्तु प्रणालियों के लिए प्रेरणा के रूप में किया गया था। सबसे अच्छी डिग्री के लिए, प्राथमिक वास्तु उपकरण एक एम्बेडेड डिवाइस प्रारूप में स्थित महत्वपूर्ण कारकों को प्रस्तुत करने के लिए उपयोग किया जाता है। एम्बेडेड सिस्टम का प्रारूप दिए गए चित्र में दिखाया गया है।

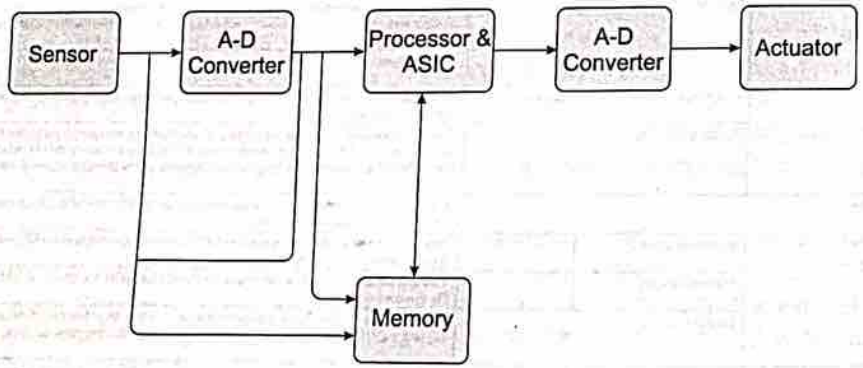


चित्र 3.10 : एम्बेडेड सिस्टम का प्रारूप

एम्बेडेड सिस्टम आर्किटेक्चर के प्रारूप को दर्शाता है, कि सभी एम्बेडेड सिस्टम बहुत ही बेहतरीन डिग्री पर एक समानता का अनुपात रखते हैं; अर्थात्, उन सभी में एक न्यूनतम एक लेयर (हार्डवेयर) या सभी लेयर (हार्डवेयर, डिवाइस सॉफ़्टवेयर और सतर्कता सॉफ़्टवेयर) होती हैं, जिसमें सभी योजक आते है । हार्डवेयर लेयर में एक एम्बेडेड बोर्ड पर रखे गए सभी महत्वपूर्ण भौतिक घटक होते हैं, जबकि सिस्टम और सतर्कता सॉफ़्टवेयर परत (Alertness software layer) में सभी सॉफ्टवेयर प्रोग्राम को रखा जाता है और एम्बेडेड गैजेट द्वारा संसाधित किया जाता है। यह संदर्भ प्रारूप मूल रूप से एक एम्बेडेड सिस्टम की संरचना का एक स्तरित (मॉड्यूलर) चित्रण है जिसमें से एक मॉड्यूलर एम्बेडेड संरचना प्राप्त की जा सकती है। जबकि लेयरिंग की अवधारणा एम्बेडेड गैजेट डिजाइन के लिए अद्वितीय नहीं है (आर्किटेक्चर सभी पीसी संरचनाओं के लिए प्रासंगिक हैं, और एक एम्बेडेड सिस्टम PC सिस्टम का एक रूप है), यह सैकड़ों के नहीं, तो बहुतो के संभावित मिश्रणों को देखने में एक उपयोगी उपकरण है। हार्डवेयर और सॉफ्टवेयर एडिटिव्स का उपयोग किया जा सकता है, जो एक एम्बेडेड डिवाइस को बनाने में उपयोग किया जाता है।

3.5 एम्बेडेड सिस्टम की सरंचना (Basic Structure of an Embedded System)

नीचे दिए गए आरेख एम्बेडेड सिस्टम की मूल संरचना को दर्शाता है—



चित्र 3.11 : एम्बेडेड सिस्टम की संरचना

सेंसर (Sensor)—यह Physical quantity को मापता है और इसे Electrical signal में बदल देता है। इस Electrical signal को Observer के द्वारा या Electrical यंत्र; जैसे—A2D Converter के द्वारा read किया जाता है। Sensor मापी गयी Quantity को Memory में स्टोर करता है।

A-D Converter—एक Analog to digital converter सेंसर के द्वारा भेजी गयी Analog signal को Digital signal में बदल देता है।

Processor & ASICs—प्रोसेसर, Output को मापने के लिए Data को प्रोसेस करता है तथा इसे Memory में स्टोर करता है।

D-A Converter—एक Digital to analog converter डिजिटल डाटा को एनालॉग डाटा में परिवर्तित करता है।

Actuator—एक Actuator D-A कनवर्टर द्वारा दी गई आउटपुट की Actual (वास्तविक) आउटपुट से तुलना करता है।

3.6 एम्बेडेड सिस्टम का महत्व (Importance of Embedded System Architecture)

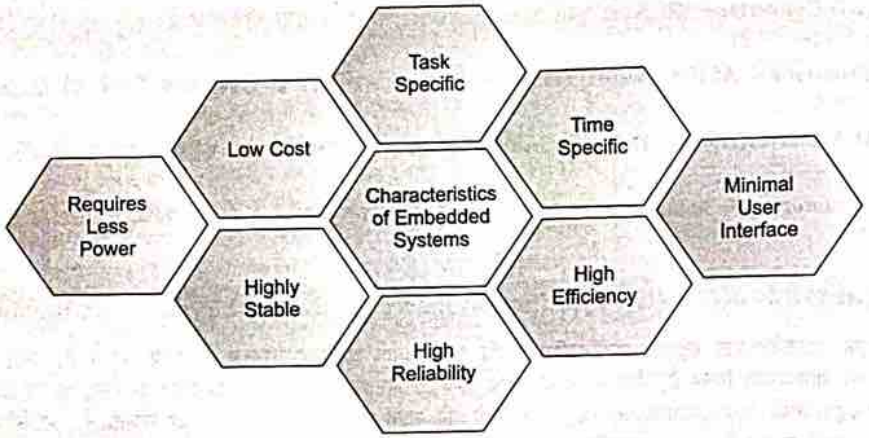
एक आर्किटेक्चरल स्ट्रक्चर इंजीनियरिंग पद्धति का उपयोग एम्बेडेड सिस्टम के लिए करता है, क्योंकि यह अधिकतम शक्तिशाली गियर है, जिसका उपयोग एक एम्बेडेड संरचना लेआउट को पहचानने के लिए या नए डिवाइस को डिजाइन करते समय सामना की गई परिस्थितियों को साफ करने के लिए किया जा सकता है। आर्किटेक्चरल तकनीक इतनी प्रभावी है, कि अनौपचारिक रूप से और त्वरित रूप से तकनीकी पृष्ठभूमि वाले लोगों के प्रसार के लिए या यहाँ तक कि एक उपकरण के रूप में कार्य करना, यहाँ तक कि असाइनमेंट की योजना बनाने में एक आधार के रूप में कार्य करना या निश्चित रूप से एक डिवाइस को डिजाइन करने की क्षमता है, क्योंकि यह वास्तव में प्रणाली की आवश्यकताओं को रेखांकित करता है, वास्तुकला उपकरण की गुणवत्ता और विभिन्न स्थितियों के नीचे उसके प्रदर्शन के अध्ययन और परीक्षण के लिए एक ठोस आधार के रूप में कार्य कर सकता है। वास्तव में, एक आर्किटेक्चर की विविध प्रणालियों को तुलनीय गुणों के साथ भाग्य के सामानों को डिजाइन करने के लिए बदला जा सकता है, परिणामस्वरूप डिजाइन की समझ का पुनः उपयोग किया जा सकता है, और भाग्य डिजाइन और विकास शुल्क की कमी की ओर अग्रसर हो सकता है।

- प्रत्येक एम्बेडेड गैजेट में एक आर्किटेक्चर होता है, चाहे वह मील की दूरी पर हो या न हो, लेकिन प्रत्येक एम्बेडेड सिस्टम में रोचक तत्व होते हैं। परिभाषिक रूप से एक आर्किटेकचर इन कारकों और उनके सम्बंधों के प्रतिनिधित्व का एक निश्चित है। दोषपूर्ण और खड़ी कीमत वाले आर्किटेक्चर होने के स्थान पर, आपको सुधार के लिए पहले से एक संरचना के लिए समय नहीं लेने के माध्यम से मजबूर किया गया था, डिजाइन का नियंत्रण निश्चित आर्किटेकचर के माध्यम से लेते हैं।
- क्योंकि एक एम्बेडेड आर्किटेकचर विविध विचारों को कैप्चर करता है, जिससे सिस्टम का प्रतिनिधित्व हो सकता है, यह सभी प्रमुख कारकों को समझने में एक लाभदायक उपकरण है। एक एम्बेडेड डिवाइस में कोई भी कारक वैक्यूम में काम नहीं करता है। डिवाइस के अंदर प्रत्येक विवरण कुछ फैशन में कुछ अलग विवरण के साथ कार्य करता है। किसी तत्व की प्रदान की गई कार्यक्षमता, समग्र प्रदर्शन और उसके आगे, 'पता चलता है' के बिना, यह निर्धारित करना अलग होगा कि गैजेट वास्तविक वैश्विक में उदाहरणों के प्रसार के नीचे कैसे व्यवहार कर सकता है।

3.7 एम्बेडेड सिस्टम की विशेषताएँ (Characteristics of an Embedded System)

एक एम्बेडेड सिस्टम की महत्वपूर्ण विशेषताएँ निम्नलिखित हैं—

- इसमें वास्तविक समय प्रदर्शन की आवश्यकता होती है।
- इसकी उच्च उपलब्धता और विश्वसनीयता होनी चाहिए।
- इसमें एक वास्तविक समय ऑपरेटिंग सिस्टम के आसपास विकसित किया गया।



चित्र 3.12 : एम्बेडेड सिस्टम की विशेषताएँ

- यह आसान और एक डिस्क रहित ऑपरेशन, ROM बूट होता है।
- इसको एक विशिष्ट कार्य के लिए बनाया गया है।
- यह इनपुट और आउटपुट डिवाइस को जोड़ने के लिए बाह्य उपकरणों से जुड़ा होना चाहिए।
- यह उच्च विश्वसनीयता और स्थिरता प्रदान करता है।
- इसे न्यूनतम उपयोगकर्ता इंटरफ़ेस की आवश्यकता होती है।
- इसमें सीमित मेमोरी, कम लागत, कम बिजली की खपत होती है।
- इसे कम्प्यूटर में किसी भी द्वितीयक मेमोरी की आवश्यकता नहीं होती है।

1. **एकल कार्य** (Single-Functioned)—यह सिस्टम एक विशेष कार्य करता है और उस कार्य को बार-बार करता है। उदाहरण के लिए—एक Pager हमेशा पेजर की तरह कार्य करेगा।
2. **Reactive और Real time**—बहुत-से एम्बेडेड सिस्टम को सिस्टम के Environment में Changes (बदलाव) होने पर लगातार React करना चाहिए और देरी के बिना Real time में परिणाम को Compute करना चाहिए।

   उदाहरण के लिए हम Car cruise controller को लेते है; यह लगातार Speed और Break Sensor की निगरानी करता है और React (प्रतिक्रिया) करता है। इसे सीमित समय में बार-बार Acceleration या De-acceleration को Compute करना चाहिए। यदि compute करने में देरी हो जाती है तो Car का Control फेल हो सकता है।
3. **माइक्रोप्रोसेसर पर आधारित** (Micro-processors Based)—ये माइक्रोप्रोसेसर या माइक्रोकंट्रोलर पर आधारित होने चाहिए।
4. **Memory**—इसमें एक मेमोरी होनी चाहिए, क्योंकि इसका सॉफ्टवेयर ROM में Embed रहता है। इसे कम्प्यूटर में किसी भी Secondary memory की जरूरत नहीं होती है।
5. **संयोजन** (Connected)—इसमें इनपुट और आउटपुट डिवाइस को कनेक्ट करने के लिए Connected peripheral होना चाहिए।
6. **हार्डवेयर-सॉफ्टवेर सिस्टम** (Hardware-Software System)—सॉफ्टवेयर का उपयोग अधिक Features और Flexibility के लिए किया जाता है तथा हार्डवेयर का उपयोग Performance और Security के लिए किया जाता है।

7. इनमें बहुत ही कम या कोई User Interface (UI) नहीं होता है। एक पूरी तरह से Automatic वाशिंग मशीन स्वयं ही काम करती है जब उसे Set कर दिया जाता है और कार्य समाप्त होने के बाद यह स्वयं बंद हो जाती है।
8. अधिकतर Embedded system छोटे आकार के होते हैं, कम Power के साथ कार्य कर सकते हैं और बहुत महंगे भी नहीं होते हैं।
9. एम्बेडेड सिस्टम को Users के द्वारा Change या Upgrade नहीं किया जा सकता है। इसलिए ये Reliable और Stable होने चाहिए और ये बिना किसी कठिनाई के लम्बे समय तक कार्य करते रहने चाहिए, जिससे कि उपयोगकर्ता को कोई मुश्किल ना आये।

3.8 एम्बेडेड सिस्टम में प्रयुक्त महत्वपूर्ण शब्दावली (Important Terminologies used in Embedded System)

अब इस एम्बेडेड सिस्टम ट्यूटोरियल में हम एम्बेडेड सिस्टम में उपयोग किए जाने वाले कुछ महत्वपूर्ण शब्दों के बारे में जानेंगे।

विश्वसनीयता (Reliability)—जब रन समय के दौरान फ़ंक्शन महत्वपूर्ण होता है, तो सिस्टम की उत्तरजीविता सम्भावना का माप होती है ।

दोष सहिष्णुता (Fault-Tolerance)—दोष-सहिष्णुता दोष की उपस्थिति में Survive करने के लिए एक कम्प्यूटर प्रणाली की क्षमता होती है।

रियल टाइम (Real-Time)—एम्बेडेड सिस्टम को विभिन्न समय और अन्य बाधाओं को पूरा करना होता है । वे बाहरी दुनिया के वास्तविक समय के प्राकृतिक व्यवहार द्वारा उस पर लगाए गए हैं। उदाहरण के लिए, आने वाले मिसाइल हमलों पर नज़र रखने वाले एयरफोर्स विभाग को कठिन वास्तविक समय सीमा के कारण अपने जवाबी हमले की सटीक गणना और योजना करनी चाहिए। अन्यथा, यह नष्ट हो जाएगा।

पोर्टेबिलिटी (Portability)—पोर्टेबिलिटी विभिन्न वातावरणों में एक ही एम्बेडेड सॉफ़्टवेयर का उपयोग करने में आसानी का एक उपाय है। इसे एप्लिकेशन प्रोग्राम लॉजिक और निम्न-स्तरीय सिस्टम इंटरफेस के बीच सामान्यीकृत अमूर्तताओं की आवश्यकता होती है।

3.9 एम्बेडेड सिस्टम के लाभ (Advantage of Embedded System)

इसके लाभ निम्नलिखित हैं—

- इसे आसानी से Customize किया जा सकता है।
- यह बहुत ही कम Power को Consume करता है।
- इसका मूल्य (Cost) बहुत ही कम होता है।
- इसकी Performance बहुत अच्छी होती है।
- ये Systems बहुत अधिक Stable तथा Reliable होते हैं।
- एम्बेडेड सिस्टम आकार में बहुत-ही छोटे होते है, इसलिए इन्हें कहीं भी ले जाया और लोड किया जा सकता है।
- ये बहुत तीव्र (Fast) होते हैं।
- ये Product की Quality को बेहतर बनाते हैं।

3.10 एम्बेडेड सिस्टम से हानियाँ (Disadvantage of Embedded System)

इसकी हानियाँ निम्नलिखित हैं—

- एक बार इन्हें Configure करने के बाद बदला नहीं जा सकता है और इन्हें Users स्वयं Upgrade या Change नहीं कर सकते।
- इनका रख-रखाव बहुत मुश्किल होता है और इन Systems की Files का Backup लेना भी कठिन होता है।
- इन Systems के लिए Troubleshooting बहुत कठिन है। एक System से दूसरे System में Data को ट्रान्सफर करना भी कठिन कार्य है।
- चूँकि ये केवल एक विशेष कार्य के लिए बनाए जाते है, इसलिए इनका हार्डवेयर सिमित होता है।

3.11 एम्बेडेड सिस्टम के अनुप्रयोग (Applications of Embedded System)

इनका प्रयोग निम्नलिखित Real life जगहों पर किया जाता है—

1. **Consumer Electronics**—टेलीविजन, डिजिटल कैमरा, Computer printers, विडियो गेम कंसोल, तथा PS4 आदि में इनका उपयोग किया जाता है।
2. **Household Appliance में**—रेफ्रिजरेटर; वॉशिंग मशीन, माइक्रोवेव ओवन, एयर कंडीशनर आदि में।
3. **Medical Equipment में**—Scanners जैसे—MRI, CT स्कैन के लिए; ECG मशीन—रक्तचाप और हृदय की धड़कन की निगरानी के लिए उपकरण।
4. **ऑटोमोबाइल में**—ईंधन इंजेक्शन प्रणाली, एंटी-लॉक ब्रेकिंग सिस्टम, संगीत और मनोरंजन प्रणाली, एयर-कंडीशनर को नियंत्रित करने आदि में इनका प्रयोग किया जाता है।
5. **उद्योगों में**—असेम्बली लाइन्स, फीडबैक के लिए, डाटा कलेक्शन के लिए इन सिस्टम का उपयोग किया जाता है।
6. **Aerospace में**—Navigation, GPS आदि में।
7. **Communication में**—Routers, satellite में इनका प्रयोग होता है।

प्रश्नावली

1. एम्बेडेड सिस्टम के बारे में संक्षिप्त विवरण लिखिए।
2. एम्बेडेड सिस्टम के इतिहास की संक्षिप्त व्याख्या कीजिए।
3. एम्बेडेड सिस्टम की संरचना के बारे में बताइए।
4. एम्बेडेड सिस्टम की कार्य प्रणाली (Functional structure) की विस्तृत व्याख्या कीजिए।
5. एम्बेडेड सिस्टम के मुख्य भाग के बारे में समझाइये।
6. एम्बेडेड माइक्रोकंट्रोलर से आप क्या समझते हैं?
7. एम्बेडेड सिस्टम के विभिन्न प्रकार लिखिए।
8. एम्बेडेड सिस्टम में प्रोसेसर के दो मुख्य भाग कौन-से होते हैं?
9. एम्बेडेड सिस्टम के विभिन्न प्रोसेसर के बारे में बताइए।

❑

एम्बेडेड ऑपरेशन सिस्टम
(Embedded Operating Systems)

SYLLABUS

Introduction, real-time operating system, factors affecting embedded system, applications of embedded system, embedded systems characteristics and features.

4.1 परिचय (Introduction)

आज के उत्पाद विकास चक्र तेजी से जटिल होते जा रहे हैं। उपलब्ध विकास के आवश्यक लक्षणों का समय के साथ विस्तार हो रहा है, डेवलपर्स को कम समय में अधिक करने के तरीके खोजने की आवश्यकता है। यह कार्य प्रबंधन और संसाधन साझाकरण में दक्षता हासिल करने के लिए एक रियल टाइम ऑपरेटिंग सिस्टम (RTOS) का उपयोग करके किया जा सकता है।

4.2 रियल टाइम ऑपरेटिंग सिस्टम [Real Time Operating System] (RTOS)

RTOS एक सॉफ्टवेयर का एक भाग है, जिसे केंद्रीय प्रसंस्करण इकाई (Central Processing Unit (CPU)) के समय को कुशलतापूर्वक प्रबंधित करने के लिए बनाया गया है। यह विशेष रूप से एम्बेडेड सिस्टम के लिए प्रासंगिक है जब समय महत्वपूर्ण होता है।

ऑपरेटिंग सिस्टम, जैसे—विंडोज और RTOS के बीच मुख्य अंतर बाह्य घटनाओं का प्रतिक्रिया समय होता है, जो अक्सर एम्बेडेड सिस्टम में पाया जाता है। एक साधारण OS उन घटनाओं के लिए एक गैर-निर्धारक प्रतिक्रिया प्रदान करता है, जिनके सम्बन्ध में कोई गारंटी नहीं होती है, कि उन्हें कब संसाधित (Processed) किया जाएगा। RTOS एक वास्तविक समय प्रतिक्रिया और अत्यधिक नियतात्मक प्रतिक्रिया प्रदान करता है। विंडोज़ या लिनक्स जैसे OS के लिए उपयोग किए जाने वाले डेवलपर्स एक एम्बेडेड RTOS की विशेषताओं से काफी परिचित होते है। उन्हें सीमित मेमोरी वाले सिस्टम में चलाने के लिए और रीसेट किए जाने की आवश्यकता के बिना अनिश्चित काल तक संचालित करने के लिए डिज़ाइन किया गया है। क्योंकि RTOS को त्वरित रूप से घटनाओं का उत्तर देने और भारी भार के तहत प्रदर्शन करने के लिए बनाया गया है, इसलिए यह अन्य OS की तुलना में बड़े कार्यों में धीमा हो सकता है। RTOS को इस बात के लिए महत्व दिया जाता है, कि वह कितनी जल्दी प्रतिक्रिया दे सकता है और उसमें उन्नत शेड्यूलिंग एल्गोरिदम (Advanced scheduling algorithm) प्रमुख घटक है।

एम्बेडेड सिस्टम की समय-महत्वपूर्णता हार्ड-रियल टाइम एयरक्राफ्ट सुरक्षा प्रणालियों के माध्यम से नरम-वास्तविक समय वॉशिंग मशीन नियंत्रण प्रणालियों से भिन्न होती है। बाद की स्थितियों में वास्तविक समय की आवश्यकताओं को पूरा करने की मौलिक मांग केवल तभी की जा सकती है, जब OS अनुसूचक के व्यवहार (OS scheduler's behaviour) की सटीक भविष्यवाणी की जा सकती है। कई ऑपरेटिंग सिस्टम एक साथ कई कार्यों को निष्पादित करने का आभास देते हैं, लेकिन यह मल्टी-टास्किंग एक भ्रम है। एक Single Core Processor केवल किसी एक समय में निष्पादन का एक ही थ्रेड चला सकता है। ऑपरेटिंग सिस्टम का शेड्यूलर तय करता है कि कौन-सा

प्रोग्राम, या थ्रेड कब चलाया जाए। तेजी से थ्रेड के बीच स्विच करके यह एक साथ मल्टीटास्किंग का भ्रम प्रदान करता है। RTOS अनुसूचक का लचीलापन प्राथमिकताओं को संसाधित करने के लिए एक व्यापक दृष्टिकोण को सक्षम बनाता है, हालांकि एक RTOS अनुप्रयोगों के बहुत संकीर्ण सेट पर केंद्रित होता है। RTOS शेड्यूलर को न्यूनतम इंटरप्ट लेटेंसी और न्यूनतम थ्रेड स्विचिंग ओवरहेड देना चाहिए। यह वही है, जो समय-महत्वपूर्ण एम्बेडेड सिस्टम के लिए RTOS को इतना प्रासंगिक बनाता है।

1. एक वास्तविक समय ऑपरेटिंग सिस्टम (RTOS) एक ऑपरेटिंग सिस्टम है, जो एक निर्दिष्ट समय बाधा के भीतर एक निश्चित क्षमता की गारंटी देता है।
2. OS एक सिस्टम प्रोग्राम है, जो एप्लिकेशन प्रोग्राम और कम्प्यूटर सिस्टम (हार्डवेयर) के बीच एक इंटरफेस प्रदान करता है।
3. उन अनुप्रयोगों पर जहाँ निर्भरता है कि किसी निश्चित समय सीमा से पहले एक निश्चित कार्य समाप्त हो जाएगा ठीक उसी तरह जैसे कि सही परिणाम प्राप्त करना है ।
4. समय सीमा को पूरा करने के अलावा, RTOS को अप्रत्याशित घटनाओं का पूर्वानुमान लगाने और कई घटनाओं को समवर्ती रूप से संसाधित करने में सक्षम होना चाहिए।

एक रियल-टाइम ऑपरेटिंग सिस्टम (RTOS) एक कंप्यूटिंग वातावरण है, जो एक विशिष्ट समय अवधि में इनपुट पर प्रतिक्रिया करता है। एक वास्तविक समय सीमा इतनी कम हो सकती है कि सिस्टम प्रतिक्रिया तात्कालिक दिखाई देती है। 'वास्तविक समय' आउटपुट का वर्णन करने के लिए वास्तविक समय कम्प्यूटिंग का भी उपयोग किया गया है, जिसमें एक लम्बी लेकिन निश्चित समय सीमा होती है। वास्तविक समय और मानक ऑपरेटिंग सिस्टम के बीच अंतर सीखना उतना ही आसान है, जितना कम्प्यूटर गेम में स्वयं की कल्पना करना। खेल में आपके द्वारा की जाने वाली प्रत्येक क्रिया उस वातावरण में चलने वाले प्रोग्राम की तरह होती है। एक गेम, जिसमें इसके वातावरण के लिए एक वास्तविक समय ऑपरेटिंग सिस्टम होता है, वह आपके शरीर के विस्तार की तरह महसूस कर सकता है, क्योंकि आप एक विशिष्ट 'समयांतराल में' कार्रवाई के लिए आपके अनुरोध और आपके कम्प्यूटर के ध्यान देने योग्य निष्पादन के बीच का समय को गिन सकते हैं। एक मानक ऑपरेटिंग सिस्टम, हालांकि, असंतुष्ट महसूस कर सकता है, क्योंकि अंतराल समय अविश्वसनीय होता है। समय की विश्वसनीयता प्राप्त करने के लिए, वास्तविक समय के कार्यक्रमों और उनके ऑपरेटिंग सिस्टम के वातावरण को किसी और चीज से पहले समय सीमा के वास्तविकरण को प्राथमिकता देनी चाहिए। गेमिंग उदाहरण में, यह प्रतिक्रिया समय और दृश्य प्रभाव संघर्ष के दौरान गिरा फ्रेम या कम दृश्य गुणवत्ता का परिणाम हो सकता है।

4.2.1 रियल टाइम करनल (Real-Time Kernel)

रियल-टाइम OS का हृदय (और हर OS का हृदय, इस बात के लिए) करनल है। करनल एक ऑपरेटिंग सिस्टम का केंद्रीय कोर है, और यह सभी OS Jobs का ध्यान रखता है।

1. बूटिंग (Booting)
2. कार्य निर्धारण (Task Scheduling)
3. स्टैण्डर्ड फंक्शन लाइब्रेरीज (Standard Function Libraries)

एक एम्बेडेड सिस्टम में अक्सर करनल सिस्टम को बूट, पोर्ट और ग्लोबल डाटा आइटम को Initialize करता है। फिर, यह शेड्यूलर को शुरू करता है और किसी भी हार्डवेयर टाइमर को तुरंत शुरू करता है, जिसे शुरू करने की आवश्यकता है। अंततः करनल मूल रूप से मेमोरी से बाहर हो जाता है (लाइब्रेरी फ़ंक्शंस को छोड़कर, यदि कोई हो), और शेड्यूलर चाइल्ड कार्यों को चलाना शुरू कर देता है।

मानक फंक्शन लाइब्रेरी—एक एम्बेडेड सिस्टम में मेमोरी फंक्शन लाइब्रेरी की बनाए रखने में पर्याप्त होती है। यदि फंक्शन शामिल होने जा रहे हैं, तो उन्हें छोटा और महत्वपूर्ण होना चाहिए।

4.2.2 बेसिक कर्नेल सर्विस (Basic Kernel Services)

करनल जो एक ऑपरेटिंग सिस्टम का एक प्रमुख भाग है, जो प्रोसेसर पर चलने वाले सॉफ़्टवेयर के लिए सबसे मुख्य सेवाएं प्रदान करता है। रीयल-टाइम ऑपरेटिंग सिस्टम ('RTOS') का 'करनल' एक 'एब्स्ट्रेक्शन लेयर' प्रदान करता है जो एप्लिकेशन सॉफ़्टवेयर से प्रोसेसर (या प्रोसेसर के सेट) के हार्डवेयर विवरण को छुपाता है, जिस पर एप्लिकेशन सॉफ़्टवेयर कार्य करता है।

विशिष्ट डिजाइनों में एक कार्य में तीन स्टेट होते हैं—

1. Running (executing on the CPU);
2. Ready (ready to be executed);
3. Blocked (waiting for an event, I/O for example).

अधिकांश कार्य अधिकांश समय अवरुद्ध या तैयार होते हैं, क्योंकि केवल एक कार्य प्रति सीपीयू एक समय में चल सकता है। तैयार पंक्ति (Queue) में आइटम्स की संख्या बहुत भिन्न हो सकती है, यह इस बात पर निर्भर करता है, कि सिस्टम को कितने कार्यों की आवश्यकता है और सिस्टम किस प्रकार का उपयोग करता है। सरल गैर-पूर्व-खाली लेकिन अभी भी मल्टीटास्किंग सिस्टम पर एक कार्य को सीपीयू पर अपना समय अन्य कार्यों के लिए छोड़ना पड़ता है, जिससे तैयार कतार स्टेट में निष्पादित होने के लिए तैयार कार्यों में अधिक से अधिक संख्या में हो सकती है। समय-समय पर अनुसूचक में तैयार सूची डाटा संरचना को अनुसूचक के महत्वपूर्ण खंड में बिताए गए सबसे खराब समय की लम्बाई को कम करने के लिए डिज़ाइन किया गया है, जिसके दौरान पूर्व-उत्सर्जन (Pre-emption) निषिद्ध है, और कुछ मामलों में सभी व्यवधान अक्षम हैं। लेकिन डाटा संरचना भी उन कार्यों की अधिकतम संख्या पर निर्भर करती है, जो तैयार सूची में हो सकते हैं। यदि तैयार सूची में कुछ कार्यों से अधिक नहीं होते हैं, तो तैयार कार्यों की दोगुनी लिंक की गई सूची इष्टतम है। यदि तैयार सूची में केवल कुछ कार्य होते हैं, लेकिन कभी-कभी अधिक होते हैं, तो सूची को प्राथमिकता से क्रमबद्ध किया जाना चाहिए। इस तरह, चलाने के लिए सर्वोच्च प्राथमिकता वाले काम को खोजने के लिए पूरी सूची के माध्यम से पुनरावृत्ति की आवश्यकता नहीं होती है। किसी कार्य को सम्मिलित करने के लिए फिर आवश्यकता होती है ।

इस खोज के दौरान पूर्व-उत्सर्जन को रोकने के लिए देखभाल नहीं की जानी चाहिए। लंबे समय तक महत्वपूर्ण खंडों को छोटे टुकड़ों में विभाजित किया जाना चाहिए। यदि कोई अवरोध उत्पन्न होता है, जो उच्च प्राथमिकता वाले कार्य को कम प्राथमिकता वाले कार्य के सम्मिलन के दौरान तैयार करता है, तो उच्च प्राथमिकता वाले कार्य को डाला जा सकता है और निम्न प्राथमिकता वाले कार्य को सम्मिलित करने से पहले तुरंत चलाया जा सकता है। आलोचनात्मक प्रतिक्रिया समय, जिसे फ्लाई बैक टाइम भी कहते हैं, वह समय है, जब किसी नए तैयार कार्य को पंक्तिबद्ध करने और चलने के लिए सर्वोच्च प्राथमिकता वाले कार्य को पुनर्स्थापित करने के लिए किया जाता है। एक RTOS में एक नया कार्य तैयार करने के लिए प्रति-पंक्ति प्रविष्टि में 3 से 20 निर्देश होते हैं, और सर्वोच्च प्राथमिकता वाले तैयार कार्य की बहाली में 5 से 30 इंस्ट्रक्शन होंगे। अधिक उन्नत प्रणालियों में, वास्तविक समय के कार्य कंप्यूटिंग संसाधनों को कई गैर-वास्तविक समय के कार्यों के साथ साझा करते हैं, और तैयार सूची मनमाने ढंग से लंबी हो सकती है। ऐसी प्रणालियों में एक शेड्यूल की गई सूची के रूप में कार्यान्वित एक अनुसूचक तैयार सूची अपर्याप्त होगी।

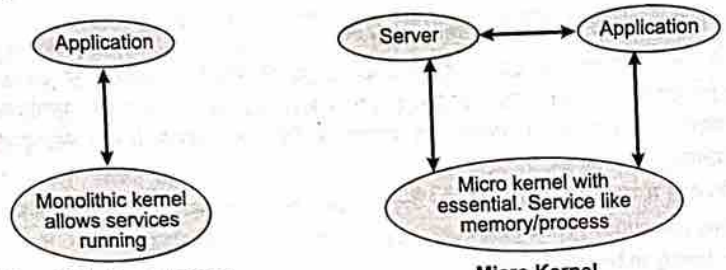
4.2.3 करनल के प्रकार (Types of Kernel)

करनल की बनावट के आधार पर, करनल को मोनोलिथिक और माइक्रोलिपिक में वर्गीकृत किया जा सकता है।

1. मोनोलिथिक करनल—आर्किटेक्चर में सभी करनल सेवा करनल स्पेस में रन करती है अर्थात् सभी करनल मॉड्यूल सिंगल करनल थ्रेड के तहत एक ही मेमोरी स्पेस के साथ रन करते हैं।

मोनोलिथिक करनल का दोष यह है, कि करनल मॉड्यूल में किसी भी त्रुटि या विफलता से पूरा करनल अनुप्रयोग के दुर्घटनाग्रस्त हो जाता है।

Ex : LINUX,SOLARIS,MS-DOS



Monolithic Kernel Model

Micro Kernel

चित्र 4.1 : करनल

2. माइक्रो करनल—डिज़ाइन करनल में केवल OS सेवाओं का आवश्यक सेट होता है। बाकी OS सेवाओं को 'सर्वर' के रूप में ज्ञात कार्यक्रमों में प्रत्यारोपित किया जाता है, जो उपयोगकर्ता स्पेस में चलता है।

The essential services of the Micro kernel are :

Memory management, Process management, Timer system, Interrupt handler

Ex. OS , MACH , ONX , MINIX3

4.3 ऑपरेटिंग सिस्टम के प्रकार (Types of Operating System)

करनल और करनल सेवाओं के प्रकार के आधार पर और कम्प्यूटिंग सिस्टम OS के प्रकार को दो वर्गों में वर्गीकृत किया गया है

1. सामान्य प्रयोजन ऑपरेटिंग सिस्टम [GPOS]
2. रियल टाइम ऑपरेटिंग सिस्टम [RTOS]

4.3.1 सामान्य प्रायोजन ऑपरेटिंग सिस्टम (General Purpose Operating System[GPOS])

सामान्य कम्प्यूटिंग सिस्टम में तैनात OS को सामान्य उद्देश्य OS के रूप में संदर्भित किया जाता है। ऐसे OS की करनल अधिक सामान्यीकृत होती है और इसमें जेनेरिक अनुप्रयोगों के निष्पादन के लिए आवश्यक सभी प्रकार की सेवाएँ शामिल होती हैं। GPOS व्यवहार में गैर-नियतात्मक हैं।

Ex : PC/Desktop system, Windows XP & MS-DOS.

4.3.2 वास्तविक समय ऑपरेटिंग सिस्टम (Real Time Operating System [RTOS])

RTOS की कोई सार्वभौमिक परिभाषा उपलब्ध नहीं है। RTOS एक OS है, जिसका उद्देश्य वास्तविक समय के अनुप्रयोग अनुरोधों की सेवा करना है। यह डाटा को संसाधित करने में सक्षम होना चाहिए, क्योंकि यह देरी के बिना आता है। प्रसंस्करण समय की आवश्यकताओं को सेकंड के दसवें भाग में मापा जाता है।

Ex : Windows CE, UNIX, VxWorks, Micros/OS-II Real Time Kernel :

RTOS को पारम्परिक OS करनल के पूरक के रूप में रियल टाइम करनल कहा जाता है। करनल अत्यधिक विशिष्ट है और इसमें उपयोगकर्ता एप्लिकेशन को चलाने के लिए आवश्यक सेवाओं का न्यूनतम सेट है।

Real Time Kernel के मुख्य कार्य

1. कार्य / प्रक्रिया प्रबंधन (Task/Process Management)
2. कार्य / प्रक्रिया निर्धारण (Task/Process Scheduling)
3. कार्य / प्रक्रिया तुल्यकालन (Task/Process Synchronization)

4. त्रुटि / अपवाद हैंडलिंग (Error/Exception Handling)
5. मेमोरी प्रबंधन (Memory Management)
6. बाधा से निपटना (Interrupt Handling)
7. समय प्रबंधन (Time Management)

1. कार्य / प्रक्रिया प्रबंधन (Task/Process Management)—कार्यों के लिए मेमोरी स्पेस सेट करने, कार्य कोड को मेमोरी स्पेस में लोड करने, सिस्टम संसाधनों को वितरित करने, कार्य और कार्य / प्रक्रिया समाप्ति / विलोपन के लिए एक TAsk Control Block (TCB) स्थापित करने से सम्बंधित है।

TCB—टास्क कंट्रोल ब्लॉक का उपयोग किसी कार्य के लिए सूचना रखने के लिए किया जाता है। TCB में सूचना का निम्नलिखित सेट होता है—

टास्क आईडी—कार्य की पहचान संख्या

टास्क स्टेट—कार्य की वर्तमान स्थिति

कार्य का प्रकार—यह बताता है, कि इस कार्य का प्रकार क्या है, कार्य कठिन वास्तविक समय या नरम वास्तविक समय या पृष्ठभूमि कार्य हो सकता है।

कार्य की प्राथमिकता—कार्य की प्राथमिकता वाले कार्य के लिए 1 = 1

कार्य की संदर्भ सूचक—संदर्भ सूचक, संदर्भ को बचाने के लिए सूचक

टास्क मेमोरी पॉइंटर—कोड मेमोरी, डाटा मेमोरी और पॉइंटर्स की ओर संकेत करता है

कार्य के लिए स्टैक मेमोरी—

टास्क सिस्टम रिसोर्स पॉइंटर्स—कार्य द्वारा प्रयुक्त सिस्टम रिसोर्स को इंगित करता है

टास्क पॉइंटर्स—अन्य TCB के लिए सूचक

अन्य पैरामीटर—अन्य प्रासंगिक कार्य पैरामीटर

TCB पैरामीटर कार्य प्रबंधन कार्यान्वयन के आधार पर विभिन्न गुठली में भिन्न होते हैं

कार्य बनाने के लिए एक कार्य के लिए TCB बनाता है।

2. कार्य / प्रक्रिया निर्धारण (Task/Process Scheduling)—विभिन्न कार्यों / प्रक्रिया के बीच CPU को साझा करना। शेड्यूलर करनल के अनुप्रयोग, कार्य शेड्यूलिंग को संभालता है। शेड्यूलर कुछ भी नहीं है, लेकिन एक एल्गोरिथम सभी को लागू किया गया है, जो एक निर्धारक व्यवहार प्रदान करने के लिए कार्य के कुशल और इष्टतम निर्धारण करता है।

3. कार्य / प्रक्रिया तुल्यकालन (Task/Process Synchronization)—सिस्टम समवर्ती पहुँच के साथ एक संसाधन जो विभिन्न कार्यों के बीच कई कार्यों और संचार साझा किया जाता है।

4. त्रुटि / अपवाद हैंडलिंग (Error/Exception Handling)—कार्यों के निष्पादन के दौरान हुई त्रुटियों को पंजीकृत करने और संभालने से संबंधित कार्य। **Ex :** अपर्याप्त स्मृति, समय बहिष्कृत, Dead lock, Dead line, बस त्रुटि, शून्य से विभाजित, अज्ञात अनुदेश निष्पादन। त्रुटियां और अपवाद दो स्तरों की सेवाओं में हो सकते हैं * करनल स्तर सेवा * कार्य स्तर।

5. मेमोरी प्रबंधन (Memory Management)—RTOS, GPOS द्वारा उपयोग की जाने वाली सामान्य डायनामिक मेमोरी आवंटन तकनीक के बजाय Based ब्लॉक आधारित मेमोरी 'आवंटन तकनीकों का उपयोग करता है।

RTOS करनल निश्चित आकार की डायनामिक मेमोरी के ब्लॉक का उपयोग करता है और ब्लॉक को आधार की आवश्यकता पर किसी कार्य के लिए वितरित किया जाता है। कुछ RTOS करनल लागू करता है वर्चुअल मेमोरी अवधारणाएँ कचरा संग्रह ओवरहेड से बचती हैं।

6. बाधा से निपटना (Interrupt Handling)—**व्यवधान सिस्टम को रीयल टाइम व्यवहार** एम्बेडेड सिस्टम डिज़ाइन प्रक्रिया प्रदान करते हैं। व्यवधान प्रोसेसर को सूचित करता है, कि बाह्य उपकरण या सम्बंधित कार्य को CPU

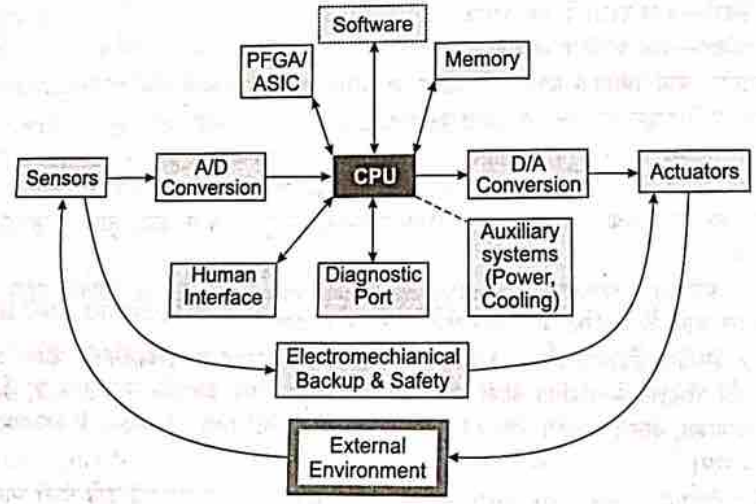
पर तत्काल ध्यान देने की आवश्यकता है। ये व्यवधान कर सकते हैं या तो सिंक्रोनस एसिंक्रोनस हो। व्यवधान जो वर्तमान में निष्पादित कार्य के साथ सिंक में होता है उसे Synchronous Interrupt के रूप में जाना जाता है। आमतौर पर सिस्टम में सिंक्रोनस श्रेणी के अंतर्गत बाधा आती है Ex : Divide by zero, मेमोरी सेगमेंटेशन एरर। एक तुल्यकालिक व्यवधान वे होते हैं, जो किसी भी कार्य के निष्पादन के किसी भी बिंदु पर होते हैं और वर्तमान में निष्पादित कार्यों के साथ सिंक में नहीं होते हैं।

4.4 RTOS का एम्बेडेड डिजाइन में प्रयोग (The use of RTOS in Embedded Designs)

बहुत-से एम्बेडेड प्रोग्रामर RTOS का उपयोग करने से कतराते हैं, क्योंकि उन्हें संदेह है, कि यह उनके एप्लीकेशन में बहुत अधिक जटिलता जोड़ता है, या यह केवल अज्ञात क्षेत्र है। RTOS को आमतौर पर अपने कर्तव्यों को निभाने के लिए CPU के 5% तक संसाधनों की आवश्यकता होती है। हालांकि इसमें हमेशा कुछ संसाधन दंड (Resource penalties) होंगे, RTOS इसके लिए सरलीकृत निर्धारणवाद, उपयोग में आसानी, हालांकि HW abstraction, विकास के समय को कम करने और आसान डिबगिंग; जैसे क्षेत्रों में बना सकता है। RTOS का उपयोग करने का मतलब है, कि आप बुनियादी कनेक्टिविटी, गोपनीयता, सुरक्षा और इसी तरह की आवश्यकता होने पर कई कार्यों को समवर्ती रूप से चला सकते हैं। RTOS आपको अपनी परियोजना की विशिष्ट आवश्यकताओं के लिए एक अनुकूलित समाधान बनाने की अनुमति देता है।

4.5 एम्बेडेड सिस्टम (Embedded Systems)

एक एम्बेडेड सिस्टम के लिए चित्र एक संभव संगठन दिखाता है।



चित्र 4.2 : An Embedded System Encompasses the CPU as well as Many other Resources

इसमें सीपीयू और मेमोरी पदानुक्रम के अतिरिक्त, विभिन्न प्रकार के इंटरफेस होते हैं, जो सिस्टम को मापने, हेरफेर करने और अन्यथा बाहरी वातावरण में कार्य करने में सक्षम बनाते हैं। यहाँ डेस्कटॉप कंप्यूटिंग के साथ कुछ अंतर हो सकते हैं—

- मानव इंटरफ़ेस फ्लशिंग लाइट के रूप में सरल या वास्तविक समय रोबोट विज़न के रूप में जटिल हो सकता है।

- डायग्नोस्टिक पोर्ट का उपयोग उस प्रणाली के निदान के लिए किया जा सकता है, जिसे नियंत्रित किया जा रहा है - न कि केवल कंप्यूटर के निदान के लिए।
- विशेष प्रयोजन क्षेत्र प्रोग्रामेबल (FPGA), अनुप्रयोग विशिष्ट (ASIC), या यहाँ तक कि Non-digital हार्डवेयर का उपयोग प्रदर्शन या सुरक्षा बढ़ाने के लिए किया जा सकता है।
- सॉफ्टवेयर में अक्सर एक निश्चित फ़ंक्शन होता है, एप्लिकेशन के लिए विशिष्ट होता है।

एम्बेडेड सिस्टम अपने अनुप्रयोगों के लिए कार्यक्षमता प्रदान करते हैं। स्प्रेडशीट, वर्ड प्रोसेसिंग और इंजीनियरिंग विश्लेषण निष्पादित करने के अतिरिक्त, एम्बेडेड सिस्टम मुख्य रूप से नियंत्रण कानूनों, परिमित राज्य मशीनों और सिग्नल प्रोसेसिंग एल्गोरिदम को निष्पादित करते हैं। उन्हें अक्सर कंप्यूटिंग और आस-पास के दोनों विद्युत प्रणालियों में दोषों का पता लगाना और प्रतिक्रिया करना चाहिए, और एप्लिकेशन-विशिष्ट उपयोगकर्ता इंटरफ़ेस उपकरणों में हेरफेर करना चाहिए।

An example of	Singl Processing	Mission Ciritical	Distributed	Small
Computting speed	1 GFLOPS	10 - 100 MIPS	1-10 MIPS	100,000 IPS
I/O Transfer Rates	1 Cb/sec	10 Mb/sec	100 Kb/sec	1 Kb/sec
Memory Size	32 - 128 MB	16 - 32 MB	1 - 16 MB	1 KB
Units Sold	10 - 500	100 - 1000	100 - 10,000	1,000,000 +
Development Cost	$ 20 – $ 100 M	$ 10 M - $ 50 M	$ 1 M - $10 M	$ 100K - $ 1 M
Lifetime	15 - 30 years	20 - 30 years	25 - 50 years	10 - 15 years
Environment	Bibration, Heat	Heat, Vibration, Lightning	Dirt, Fire	Over-voltage, Heat, Vibration
Cost Sensitivity	$ 1000	$ 100	$ 10	$ 0.05
Other constraints	Size weight, power	Size, weight	Size	Size, weight, Heat, Vibration
Safety	—	Redundancy	Mechanical Safety	—
Maintenance	Frequent repairs	Aggressive fault detection/maintenance	Scheduled maintenanced	"Never" breaks
Digital content	Digital except for signal I/O	–1/ 2 Digital	– 1/2 Digital	Signal digital chip, rest is analog/power
Certification authorities	Customer	Federal Government	Development team	Customer, Federal Government
Repair time goal	1-20 hours	30 minutes	4 min - 12 hours	1-4 hours
Initial cycle time	3-5 years	4 - 10 years	2 - 4 years	0.1 - 4 years
Product variants	1-5	5 - 20	10 - 100, 000	3-10
Engineering allocation method	Per-product budget	Per-product budget	Allocation from large pool	Demand-driven daily from small pool
Other possible example in this category	Radar/Sonar Video Medical imaging	Jet engines Manned spacecraft Nuclear power	High-rise elevators Trains/trans/subways Air conditioning	Automotive auxilliaries Consumer electronics "Smart" I/O

Table 1. Embedded Systems with Approximate Attributes.

चर्चा को अधिक ठोस बनाने के लिए, हम चार उदाहरण प्रणालियों (तालिका 1) पर चर्चा करेंगे। प्रत्येक उदाहरण वर्तमान उत्पादन में एक वास्तविक प्रणाली को चित्रित करता है, लेकिन अनुप्रयोगों के व्यापक क्रॉस-सेक्शन का प्रतिनिधित्व करने के साथ-साथ स्वामित्व के हितों की रक्षा करने के लिए थोड़ा सामान्य किया गया है। चार उदाहरण सिग्नल प्रोसेसिंग सिस्टम, मिशन क्रिटिकल कंट्रोल सिस्टम, वितरित नियंत्रण प्रणाली और छोटा उपभोक्ता इलैक्ट्रॉनिक

सिस्टम है। सिग्नल प्रोसेसिंग और मिशन क्रिटिकल सिस्टम पारंपरिक सैन्य/एयरोस्पेस एम्बेडेड सिस्टम के प्रतिनिधि हैं, लेकिन वास्तव में समय के साथ सामान्य वाणिज्यिक अनुप्रयोगों के लिए अधिक लागू हो रहे हैं। बिंदुओं को स्पष्ट करने के लिए इन चार उदाहरणों का उपयोग करते हुए, निम्नलिखित खंड एम्बेडेड सिस्टम डिज़ाइन के विभिन्न क्षेत्रों कम्प्यूटर डिज़ाइन, सिस्टम-स्तरीय डिज़ाइन, जीवन-चक्र समर्थन, व्यवसाय मॉडल समर्थन और डिज़ाइन संस्कृति अनुकूलन का वर्णन करते हैं।

डेस्कटॉप कंप्यूटिंग डिजाइन पद्धति और उपकरण का समर्थन डिजिटल सिस्टम के प्रारंभिक डिजाइन से संबंधित एक बड़ी डिग्री है। यह सुनिश्चित करने के लिए, अनुभवी डिज़ाइनर अन्य पहलुओं से परिचित हैं, लेकिन मात्रात्मक डिज़ाइन जीवन-चक्र के मुद्दों पर हाल ही में जोर देने के साथ जो कि आसानी से मात्रा निर्धारित नहीं हैं, उन्हें अनुकूलन प्रक्रिया से बाहर रखा जा सकता है। हालांकि, इस तरह के एक दृष्टिकोण एम्बेडेड सिस्टम बनाने के लिए अपर्याप्त है जो बाजार में प्रभावी रूप से प्रतिस्पर्धा कर सकते हैं। ऐसा इसलिए है, क्योंकि कई विषयों में मुद्दा यह नहीं है कि क्या एक बेहद जटिल प्रणाली का डिजाइन संभव है, बल्कि यह भी कि क्या अपेक्षाकृत मामूली प्रणाली को जीवन-चक्र की लागत और प्रभावशीलता के लिए अत्यधिक अनुकूलित किया जा सकता है।

हालांकि पारंपरिक डिजिटल डिजाइन सीएडी उपकरण एक कम्प्यूटर डिजाइनर को अधिक कुशल बना सकते हैं, वे केंद्रीय मुद्दे से निपट नहीं सकते हैं-एम्बेडेड डिजाइन सिस्टम के बारे में है, कंप्यूटर के बारे में नहीं। डेस्कटॉप कंप्यूटिंग में डिजाइन अक्सर सबसे तेज सीपीयू के निर्माण पर केंद्रित होता है, फिर अधिकतम कंप्यूटिंग गति के लिए आवश्यक रूप से इसका समर्थन करता है। एम्बेडेड सिस्टम में बाह्य इंटरफेस (सेंसर, एक्ट्यूएटर) और कंट्रोल या सिक्वेंसिंग एल्गोरिदम का संयोजन प्राथमिक महत्व है। सीपीयू बस उन कार्यों को लागू करने के एक तरीके के रूप में उपस्थित है। निम्नलिखित प्रयोग इस बिंदु को स्पष्ट करने के लिए करने चाहिए: एक कमरे में रहने वाले लोगों से पूछें कि पर्सनल कंप्यूटर या उनके द्वारा उपयोग किए जाने वाले कार्य केंद्र में किस प्रकार का सीपीयू है। फिर उन्हीं लोगों से पूछें कि उनकी कार में इंजन नियंत्रक के लिए सीपीयू का उपयोग किया जाता है और क्या सीपीयू प्रकार ने खरीद निर्णय को प्रभावित किया है। उच्च अंत एम्बेडेड सिस्टम में डेस्कटॉप कंप्यूटर डिजाइन के लिए उपयोग किए जाने वाले उपकरण अमूल्य हैं। हालांकि, कई एम्बेडेड सिस्टम बड़े और छोटे दोनों को अतिरिक्त आवश्यकताओं को पूरा करना चाहिए जो कि डिज़ाइन ऑटोमेशन द्वारा आमतौर पर नियंत्रित किए जाने वाले दायरे से परे हैं। ये अतिरिक्त जरूरतें विशेष कंप्यूटर डिजाइन आवश्यकताओं, सिस्टम-स्तरीय आवश्यकताओं, जीवन-चक्र समर्थन मुद्दों, व्यवसाय मॉडल संगतता और डिजाइन संस्कृति मुद्दों की श्रेणियों में आती हैं।

4.6 Computer Design Requirements

Embedded computers typically have tight constraints on both functionality and implementation. In particular, they must guarantee real time operation reactive to external events, conform to size and weight limits, budget power and cooling consumption, satisfy safety and reliability requirements, and meet tight cost targets.

4.6.1 Real Time/reactive Operation

Real time system operation means that the correctness of a computation depends in part on the time at which it is delivered. In many cases the system design must take into account worst case performance. Predicting the worst case may be difficult on complicated architectures, leading to overly pessimistic estimates erring on the side of caution. The Signal Processing and Mission Critical example systems have a significant requirement for real time operation in order to meet external I/O and control stability requirements.

Reactive computation means that the software executes in response to external events. These events may be periodic, in which case scheduling of events to guarantee performance may be possible. On the other hand, many events may be aperiodic, in which case the maximum event arrival rate must be estimated in order to accommodate worst case situations. Most embedded systems have a significant reactive component.

Design Challenge :

- Worst case design analyses without undue pessimism in the face of hardware with statistical performance characteristics, *e.g.*, cache memory.

4.6.2 Small Size, Low Weight

Many embedded computers are physically located within some larger artifact. Therefore, their form factor may be dictated by aesthetics, form factors existing in pre-electronic versions, or having to fit into interstices among mechanical components. In transportation and portable systems, weight may be critical for fuel economy or human endurance. Among the examples, the Mission Critical system has much more stringent size and weight requirements than the others because of its use in a flight vehicle, although all examples have restrictions of this type.

Design Challenges :

- Non-rectangular, non-planar geometries.
- Packaging and integration of digital, analog, and power circuits to reduce size.

4.6.3 Safe and Reliable

Some systems have obvious risks associated with failure. In mission-critical applications such as aircraft flight control, severe personal injury or equipment damage could result from a failure of the embedded computer. Traditionally, such systems have employed multiply-redundant computers or distributed consensus protocols in order to ensure continued operation after an equipment failure.

However, many embedded systems that could cause personal or property damage cannot tolerate the added cost of redundancy in hardware or processing capacity needed for traditional fault tolerance techniques. This vulnerability is often resolved at the system level as discussed later.

Design Challenge :

- Low-cost reliability with minimal redundancy.

4.6.4 Harsh Environment

Many embedded systems do not operate in a controlled environment. Excessive heat is often a problem, especially in applications involving combustion, *e.g.*, many transportation applications. Additional problems can be caused for embedded computing by a need for protection from vibration, shock, lightning, power supply fluctuations, water, corrosion, fire and general physical abuse. For example, in the Mission Critical example application the computer must function for a guaranteed, but brief, period of time even under non-survivable fire conditions.

Design Challenges :

- Accurate thermal modelling.
- De-rating components differently for each design, depending on operating environment.

4.6.5 Cost Sensitivity

Even though embedded computers have stringent requirements, cost is almost always an issue (even increasingly for military systems). Although designers of systems large and small may talk about the importance of cost with equal urgency, their sensitivity to cost changes can vary

dramatically. A reason for this may be that the effect of computer costs on profitability is more a function of the proportion of cost changes compared to the total system cost, rather than compared to the digital electronics cost alone. For example, in the Signal Processing system cost sensitivity can be estimated at approximately $1000, *i.e.*, a designer can make decisions at the $1000 level without undue management scrutiny. However, with in the small system decisions increasing costs by even a few cents attract management attention due to the huge multiplier of production quantity combined with the higher percentage of total system cost it represents.

Design Challenge :

- Variable 'design margin' to permit tradeoff between product robustness and aggressive cost optimization.

4.7 System-level Requirements

In order to be competitive in the marketplace, embedded systems require that the designers take into account the entire system when making design decisions.

4.7.1 End-product utility

The utility of the end product is the goal when designing an embedded system, not the capability of the embedded computer itself. Embedded products are typically sold on the basis of capabilities, features and system cost rather than which CPU is used in them or cost/performance of that CPU.

One way of looking at an embedded system is that the mechanisms and their associated I/O are largely defined by the application. Then, software is used to coordinate the mechanisms and define their functionality, often at the level of control system equations or finite state machines. Finally, computer hardware is made available as infrastructure to execute the software and interface it to the external world. While this may not be an exciting way for a hardware engineer to look at things, it does emphasize that the total functionality delivered by the system is what is paramount.

Design Challenge :

- Software and I/O-driven hardware synthesis as opposed to hardware-driven software compilation/synthesis.

4.7.2. System Safety and Reliability

An earlier section discussed the safety and reliability of the computing hardware itself. But, it is the safety and reliability of the total embedded system that really matters. The Distributed system example is mission critical, but does not employ computer redundancy. Instead, mechanical safety backups are activated when the computer system loses control in order to safely shut down system operation.

A bigger and more difficult issue at the system level is software safety and reliability. While software doesn't normally 'break' in the sense of hardware, it may be so complex that a set of unexpected circumstances can cause software failures leading to unsafe situations. This is a difficult problem that will take many years to address, and may not be properly appreciated by non-computer engineers and managers involved in system design decisions discusses the role of computers in system safety.

Design Challenges :

- Reliable software.

- Cheap, available systems using unreliable components.
- Electronic vs. non-electronic design tradeoffs.

4.7.3 Controlling Physical Systems

The usual reason for embedding a computer is to interact with the environment, often by monitoring and controlling external machinery. In order to do this, analog inputs and outputs must be transformed to and from digital signal levels. Additionally, significant current loads may need to be switched in order to operate motors, light fixtures and other actuators. All these requirements can lead to a large computer circuit board dominated by non-digital components.

In some systems 'smart' sensors and actuators (that contain their own analog interfaces, power switches and small CPUS) may be used to off-load interface hardware from the central embedded computer. This brings the additional advantage of reducing the amount of system wiring and number of connector contacts by employing an embedded network rather than a bundle of analog wires. However, this change brings with it an additional computer design problem of partitioning the computations among distributed computers in the face of an inexpensive network with modest bandwidth capabilities.

Design Challenge :

- Distributed system tradeoffs among analog, power, mechanical, network, and digital hardware plus software.

4.7.4. Power management

A less pervasive system-level issue, but one that is still common, is a need for power management to either minimize heat production or conserve battery power. While the push to laptop computing has produced 'low-power' variants of popular CPUs, significantly lower power is needed in order to run from inexpensive batteries for 30 days in some applications, and up to 5 years in others.

Design Challenge :

- Ultra-low power design for long-term battery operation.

4.8 Components of an Embedded System

Embedded systems are divided into two types of components :

Hardware components and Software components

Hardware components include :

Power supply :

- A power supply is an essential part of an embedded system. The power supply may be provided from a battery or an adapter. Depending on the application that the embedded system is being used in.
- A good power supply means.
- Efficiency, stable and smooth output, transient response.

Processor :

- The processor acts as the main brain in an embedded system. An embedded system can use a micro-controller or a micro-processor.

- Some of the criteria that is considered when choosing a processor are : Speed, amount of RAM and ROM, operating voltage, packaging.

Memory :

- There are usually three types of memory associated with embedded systems: (RAM and ROM are more common).

Read-Only Memory(ROM) :

- This is used to store a program. When the system is powered on, the system will get the code it requires to operate from the ROM memory.

Random Access Memory(RAM) :

- This type of memory is volatile memory and is used to store data temporarily in storage.

Electrically Erasable Programable Read-Only Memory (EEPROM) :

- This type of memory is unique, it is the least used from the three. It allows content to be erased and reprogrammed by using a high volt pule input. This is used to store the data by the program itself.

Timers-Counters :

- For applications which require a delay to function or to operate, a timer and counter is used to generate a delay for a specific time interval without affecting the normal code execution.

Communication ports :

- Embedded systems have a number of different types of communication ports to communicate with other embedded systems/devices.

Input and Output :

- To communicate and interact with these systems, some form of input is required. The input can be in the form of a user touch screen.

Application Specific Integrated Circuit (ASIC) :

- The circuit consist of a chip that is customised for a particular use.

Software components include

Assembler :

- Assembly language is converted to HEX code using this utility.

Emulator :

- Hardware or software which has a similar functionality to the target system which will be deployed. It can be considered as a replica of the target system.

Debugger :

- Allows programmers to find and solve errors when the output is trying to be achieved but fails.

Compiler :

- Converts programming language into target code that an interface can understand.
- It converts high level code to low level code like, machine code, assembly language, or object code.

4.9 Advantages of Embedded Systems

The advantage of adding embedded systems to the system environment are the following :

Small size & Specific :

- Embedded systems are specific to carry out certain and unique functions, rather than a system which incorporates many functions, this means their size and custom design will only have the necessary components for them to function.

Reduced Cost :

- Considering it is function specific, the user will be paying for a specific function that they desire, rather than many functions which are included anyways, regardless if a user asks for them, this inevitably means that cost can be reduced.

Portability :

- As the first point mentioned 'size', this also encapsulates another attribute which is advantageous, and that is portability. Portable systems include mobile phones.

Low Power Operation :

- Many applications for example in medicine require energy saving appliances that can function for hours without having to plug them back into a power supply to recharge. This useful feature allows embedded systems to be reliable when functioning.

Real Time Response :

- Embedded systems are also called real time systems, where the response to external event has to be instant. Therefore, they are beneficial for applications where the response to an external stimuli is critical. E.g: the deployment of airbags inside a car after collision for instance.

4.10 Limitations of Embedded Systems

The limitations of any particular embedded system are the specifications for which it was designed for. Some of the limitations are listed below :

Difficult to Upgrade :

- Embedded systems are hard to upgrade, this is because they are system specific, you may require to remove and produce/add a new embedded system instead, designed specifically for the upgrade being done.

Nearly not scalable :

- Carrying on from the point above, a system upgrade could means that if a system becomes more evolved, or if a working environment becomes more enhanced, then the embedded system will not be able to function as efficiently.

No Upgrades Available :

- Once the embedded system is configured and placed into functioning order, it cannot be changed, moreover this means any enhancement or any upgrade of any sort can not be executed.

Difficult Maintenance :

- Not only are embedded system difficult to maintain as they require specific hardware constantly, it is also known to be difficult to obtain back-ups of embedded files.

Limited Hardware and Troubleshooting Difficulty :

- Having a system for specific tasks isn't all beneficial of-course, this is because hardware constantly needs to be purchased, whether it was old or new, to maintain the function the system is trying to execute. Furthermore, troubleshooting and the transfer of data from one system to another can be problematic.

4.11 एम्बेडेड सिस्टम के अनुप्रयोग (Applications of Embedded Systems)

- आज, बड़े पैमाने पर और जटिल संगठन अपनी उत्पादकता और गुणवत्ता बढ़ाने के लिए एम्बेडेड सिस्टम का उपयोग कर रहे हैं।
- एम्बेडेड सिस्टम की सर्वोत्तम विशेषताओं के कारण इसका उपयोग विभिन्न प्रकार के अनुप्रयोगों में किया जाता है।

यहाँ, हम वास्तविक जीवन में एम्बेडेड सिस्टम के कुछ अनुप्रयोगों पर प्रकाश डालेंगे—

4.11.1 मेडिकल सेक्टर (Medical Sector)

चिकित्सा उद्योग में सेंसर और पर्यावरण नियंत्रण तंत्र जैसे विभिन्न चिकित्सा उपकरणों में एम्बेडेड सिस्टम का उपयोग किया जाता है।

उदाहरण—

- एमआरआई, सीटी और पीईटी स्कैनर (रेडियो फ्रीक्वेंसी पल्स और एक्स-रे का उपयोग करने के लिए)
- सोनोग्राफी (अल्ट्रासाउंड इमेजिंग)
- डिफाइब्रिलेटर (हृदय संबंधी असामान्यताओं का पता लगाने के लिए उपयोग किया जाता है)
- डिजिटल फ्लो सेंसर (रोगी की श्वसन प्रणाली की जाँच के लिए)
- रक्तचाप उपकरण (मानव शरीर में सिस्टोलिक और डायस्टोलिक दाब का पता लगाने के लिए)
- ग्लूकोज टेस्ट सेट (मानव शरीर में शर्करा के स्तर का परीक्षण करने के लिए)
- भ्रूण की निगरानी मशीन (गर्भावस्था, श्रम और प्रसव के दौरान उपयोग करने के लिए)
- पहनने योग्य उपकरण जो आपके स्वास्थ्य की निगरानी करते हैं (यह उपयोगकर्ताओं को उनकी हृदय गति, रक्तचाप, ग्लूकोज, वजन और कई अन्य मापदंडों की निगरानी करने की अनुमति देता है)

4.11.2 विनिर्माण उद्योग (Manufacturing Industry)

इन उद्योगों में विभिन्न प्रकार की मशीनों का उपयोग करने के लिए और इन मशीनों में प्रदर्शन कार्यों के आधार पर कई एम्बेडेड सिस्टम होते हैं।

उदाहरण—

- औद्योगिक रोबोट (उपकरण, भागों, परिमार्जन और अन्य सामग्री को स्थानांतरित करने के लिए)
- असेम्बली लाइन
- प्रतिक्रिया के लिए सिस्टम
- डाटा संग्रह के लिए सिस्टम

4.11.3 घरेलु उपकरण (Home Appliances)

एंबेडेड सिस्टम का उपयोग विभिन्न प्रकार के घरेलू उपकरणों में भी किया जाता है जो हमारे दैनिक जीवन में उपयोग किए जाते हैं, और हम पूरी तरह से इन वस्तुओं पर निर्भर रहते हैं।

उदाहरण—

- रेफ्रिजरेटर
- माइक्रोवेव ओवन्स
- वाशिंग मशीन
- एयर कंडीशनर

4.11.4 ऑटोमोटिव सेक्टर (Automotive Sector)

अधिकतर उपयोगकर्ता ऑटोमोटिव एम्बेडेड सिस्टम का उपयोग करते हैं, क्योंकि वे उपयोगकर्ताओं को अपनी सुरक्षा प्रणालियों के उपयोग से सुरक्षा प्राप्त करने की अनुमति देते हैं।

उदाहरण—

- एंटी-लॉक ब्रेकिंग सिस्टम (ABS)
- ट्रैक्शन कंट्रोल (TCS)
- इलैक्ट्रॉनिक स्थिरता नियंत्रण (F&Smeheer)।
- स्वचालित तापमान नियंत्रक

4.11.5 ऑटोमोबाइल सेक्टर (Automobiles Sector)

आधुनिक कारों में विभिन्न प्रकार के एम्बेडेड सिस्टम होते हैं, जो आपकी कार में उनके अनुप्रयोगों के आधार पर विभिन्न कार्य किए जाते हैं।

उदाहरण—

- क्रूज नियंत्रण
- सस्पेंशन नियंत्रण
- एयरबैग सिस्टम
- कार मनोरंजन और मल्टीमीडिया
- वाहन का पॉवर स्टीयरिंग
- बैकअप सेंसर
- नेविगेशन सिस्टम
- मोटर नियंत्रण प्रणाली
- वाहन की इलैक्ट्रॉनिक खिड़कियां
- वाहन का ईंधन इंजेक्शन प्रणाली

4.11.6 संचार सेक्टर (Telecommunication Sector)

टेलीकॉम उद्योग में एंबेडेड सिस्टम भी महत्वपूर्ण भूमिका निभाते हैं, क्योंकि वे अल्ट्रा स्पीड नेटवर्किंग क्षमताओं को बढ़ाने में मदद करते हैं।

उदाहरण—

- डाटा राउटर
- नेटवर्क स्विच
- सैटेलाइट फोन
- वायरलेस संचार
- मॉडेम
- मोबाइल कंप्यूटिंग
- नेटवर्किंग

4.11.7 कंज्यूमर इलैक्ट्रॉनिक्स (Consumer Electronics)

एंबेडेड सिस्टम का उपयोग विभिन्न प्रकार के इलैक्ट्रॉनिक उत्पादों में भी किया जाता है और इन वस्तुओं को भी हमारे दैनिक जीवन का भाग बनाया गया है, क्योंकि उन सिस्टम के बिना हम नहीं रह सकते हैं।

उदाहरण—

- टेलीविजन
- डिजिटल कैमर
- कम्प्यूटर प्रणाली, माउस, कीबोर्ड नियंत्रक, लैन कार्ड, प्रिंटर, स्कैनर

- प्रिंटर
- फैक्स मशीन
- वीडियो गेम कंसोल
- म्युजिक सिस्टम
- डिजिटल घड़ी
- डिजिटल लॉकर
- सेट टॉप बॉक्स
- होम एंटरटेनमेंट सिस्टम जैसे PS4
- विज्ञापनों के लिए डिजिटल रोलिंग डिस्प्ले
- डिशवॉशर
- थर्मोस्टैट

4.11.8 सिग्नल सिस्टम (Signal Systems)

सिग्नलिंग सिस्टम एम्बेडेड तकनीक में शामिल है, और यह आपकी यात्रा अवधि में सुरक्षा की अनुमति देता है।

उदाहरण—ट्रैफिक मॉनिटरिंग और कोलिशन अलर्ट सिस्टम

सिग्नल सिस्टम का उपयोग विभिन्न क्षेत्रों में किया जाता है, जैसे—

- राजमार्ग (कार, बस, ट्रक, और अधिक)
- रेलमार्ग (ट्रेन)
- एयरलाइंस (विमान)
- जल यात्रा (महासागर के जहाज)

4.11.9 डिफेन्स और एरोस्पेस (Defense and Aerospace)

मिसाइल गाइडेंस सिस्टम, नेविगेशन और मार्गदर्शन के लिए सिस्टम, जीपीएस, स्पेस एक्सप्लोरेशन (रोवर्स)।

4.11.10 बैंकिंग सेक्टर (Banking Sector)

बैंकिंग क्षेत्र सुरक्षा उद्देश्य के लिए विभिन्न क्षेत्रों में एम्बेडेड सिस्टम का भी उपयोग करते हैं। उदाहरण के लिए स्मार्ट कार्ड, एटीएम, एंटी-लॉक बैंकिंग सिस्टम और बहुत कुछ हैं।

4.12 एम्बेडेड सिस्टम के उदाहरण (Examples of Embedded System)

एंबेडेड सिस्टम के उदाहरणों की अंतहीन सूची है, इसलिए हम इन उदाहरणों का विस्तार से वर्णन नहीं किया जा सकता लेकिन, हम एंबेडेड सिस्टम के कुछ उदाहरणों के बारे में विस्तार से चर्चा करेंगे।

4.12.1 कैलकुलेटर (Calculator)

- कैलकुलेटर एम्बेडेड सिस्टम का मुख्य उदाहरण है, जो वास्तविक जीवन में उपयोग किया जाता है।
- हम दैनिक जीवन में कैलकुलेटर का उपयोग करते हैं, क्योंकि यह कई गणितीय और वैज्ञानिक गणितीय समस्याओं को हल करने में मदद करता है।
- इस कैलकुलेटर में हम कीपैड के माध्यम से इनपुट सम्मिलित करते हैं जो सतह पर रखा जाता है, फिर यह विभिन्न कार्यों, जैसे—जोड़, घटा, गुणा, भाग और अन्य वैज्ञानिक प्रक्रियाओं को करता है, फिर यह एलसीडी पर शुद्ध परिणाम देता है।

- आजकल वैज्ञानिक कैलकुलेटर को अधिक लोकप्रियता मिल रही है और इसमें बहुत उच्च प्रदर्शन प्रोसेसर है, इसलिए यह विभिन्न जटिल गणितीय कार्यों को निष्पादित करने में सक्षम है।

4.12.2 इंडस्ट्रियल रोबोट (Industrial Robots)

- ये रोबोट विनिर्माण संगठनों में महत्वपूर्ण भूमिका निभाते हैं।
- आजकल सभी कार्यों को स्वचालन मोड किया जा रहा है।
- औद्योगिक रोबोट के विभिन्न प्रकार होते है और प्रत्येक संस्करण अलग-अलग कार्य करते हैं।
- कुछ रोबोट एक स्थान से दूसरे स्थान पर घूमने वाले उपकरण, स्क्रैप और अन्य सामग्रियों के लिए उपयोग किए जाते हैं।
- कुछ रोबोट विभिन्न घटकों के निर्माण के लिए उपयोग किए जाते हैं, ओर दूसरी तरफ कुछ रोबोट का उपयोग बड़ी मात्रा में भागों के संयोजन के लिए किया जाता है।
- आजकल, वाणिज्यिक उद्योगों में चित्रकारी रोबोट अधिक लोकप्रिय हैं।
- इन रोबोट के उपयोग के कारण, कम श्रमिकों की आवश्यकता होती है, क्योंकि वे शुद्ध परिणाम के साथ कम समय अवधि में अपना कार्य करते हैं।
- उत्पादकता बढ़ाने के लिए, एम्बेडेड सिस्टम वाले इन रोबोट के उपयोग के कारण।
- ये रोबोट विभिन्न कार्यक्रमों द्वारा नियंत्रित होते हैं, जिन्हें रोबोट के अंदर मेमोरी में कोडित किया जाता है।

4.12.3 पर्सनल डिजिटल असस्टिंट (Personal Digital Assistant)

- पर्सनल डिजिटल असिस्टेंट भी एम्बेडेड सिस्टम का सबसे अच्छा उदाहरण है क्योंकि ये हाथ से संचालित डिवाइस; जैसे—डाटा आयोजक, मोबाइल फ़ोन, पर्सनल डिजिटल असिस्टेंट आदि हैं।
- पर्सनल डिजिटल असिस्टेंट को स्मार्ट फोन के विकास से पहले जारी किया गया था।
- उपयोगकर्ता इन उपकरणों का उपयोग करने के साथ कई कार्य करते हैं; जैसे—डाटा एकत्रित करना, फिल्में देखना, गेम खेलना, और इंटरनेट का उपयोग।
- व्यक्तिगत डिजिटल सहायक में, अपने टच स्क्रीन इंटरफेस के माध्यम से इनपुट देने और इसका डाटा मेमोरी कार्ड में है।

4.12.4 ऑटोमेटेड टेलर मशीन [(Automated Teller Machine (ATM)]

- प्रत्येक व्यक्ति एटीएम के बारे में जानता है, इस मशीन का उपयोग धन निकालने के लिए किया जाता है।
- लगभग सभी बैंकों की अपनी एटीएम मशीनें हैं, जो अलग-अलग स्थानों पर स्थित हैं।
- यह कैसे कार्य करता है- सबसे पहले आपको एटीएम मशीन में अपना एटीएम कार्ड स्वाइप करना होगा और फिर अपना पासवर्ड डालना होगा, और आपको अपना कैश मिल जाएगा।

4.12.5 आटोमेटिक वाशिंग मशीन (Automatic Washing Machine)

- वॉशिंग मशीन एम्बेडेड सिस्टम के प्रसिद्ध दैनिक जीवन उदाहरणों में से एक है।
- वॉशिंग मशीन आजकल लगभग प्रत्येक घर में है क्योंकि इसका उपयोग गंदे कपड़े धोने के लिए किया जाता है।

- आप मशीन के अंदर सभी गंदे कपड़े डालते हैं और स्टार्ट बटन को पुश करते हैं यानी आप डाटा इनपुट करते हैं।
- कपड़े के वजन को निर्धारित करने के लिए वॉशिंग मशीन जो आपने इसके अंदर रखी है
- फिर, आपको इसके अंदर साफ पानी डालना होगा।
- वॉशिंग मशीन का वाल्व इसके लोड और जल स्तर को पूरा करने के बाद बंद हो जाता है।
- जब, आपके कपड़े धोए जाते हैं, तो सारा गंदा पानी आउटलेट वाल्व के द्वारा निकाला जाता है।
- सेंसर यह निर्धारित करता है कि गंदा पानी छोड़ा गया है या नहीं, और आउटलेट वाल्व बंद हो गया है या नहीं।

4.12.6 डिजिटल कैमरा (Digital Camera)

- डिजिटल कैमरा भी एम्बेडेड सिस्टम का मुख्य उदाहरण है।
- आजकल अधिकतर हम डिजिटल कैमरे का उपयोग अपनी कई विशेषताओं के साथ करते हैं, लेकिन ये विशेषताएँ साधारण कैमरे में नहीं थीं, क्योंकि वर्तमान में डिजिटल कैमरे में एम्बेडेड सिस्टम एकीकृत है।
- इसमें डीएसपी प्रोसेसर को सक्षम करने के साथ विभिन्न घटक होते हैं।
- इसमें कैप्चर इमेज, स्टोर इमेज डाटा, और उस डाटा का प्रतिनिधित्व करने जैसे तीन कार्य हैं।
- डिजिटल कैमरा विभिन्न प्रकार की मेमोरी जैसे RAM, मेमोरी कार्ड, कंट्रोलर सहित फ्लैश मेमोरी आदि होती हैं।
- डिजिटल कैमरे में सभी छवियों को सहेजा जाता है और बिट्स के रूप में डिजिटल डाटा के रूप में प्रक्रिया प्राप्त करके उपयोग किया जाता है।
- इसलिए, छवियों को संग्रहीत करने के लिए फिल्म की आवश्यकता नहीं होती है। इस सुविधा के कारण, छवियों को स्थानांतरित करने के लिए आसानी से भंडारण क्षमता बढ़ाने के लिए आसानी से स्थानान्तरित किया जा सकता है।

4.13 एंबेडेड सिस्टम की मुख्य विशेषताएँ (Key Features of an Embedded System)

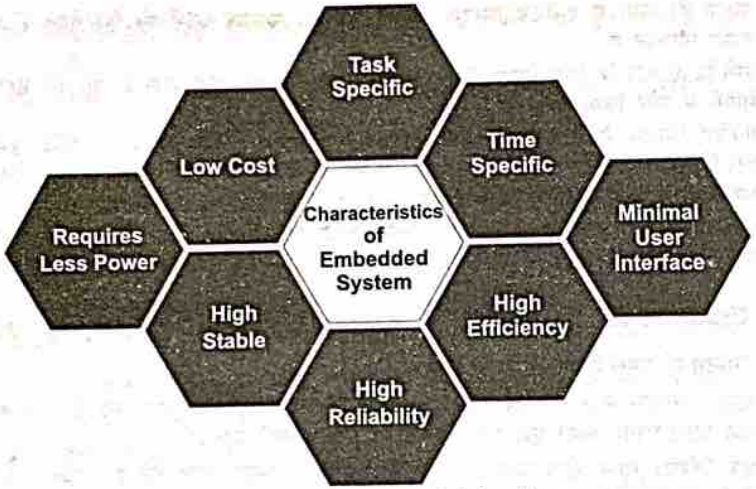
एक कंप्यूटर सिस्टम जो एक बड़े डिवाइस के भाग के रूप में फ़ंक्शन के एक विशिष्ट सेट को निष्पादित करता है, जिसे एंबेडेड सिस्टम के रूप में जाना जाता है। एंबेडेड सिस्टम उन प्रणालियों या उपकरणों में शामिल होते हैं, जिनमें हार्डवेयर और सॉफ्टवेयर का संयोजन होता है। इन प्रणालियों या उपकरणों में घरेलू इलेक्ट्रॉनिक्स, मोबाइल फोन, एमपी-3 प्लेयर या मनोरंजन उपकरण, जैसे—टीवी या म्युजिक सिस्टम शामिल हैं। एंबेडेड सिस्टम को बड़े उपकरण और मशीनों में भी लगाया किया जाता है, जैसे—कारखाना रोबोट, ट्रैफ़िक लाइट, और यहां तक कि विमान के लिए नियंत्रण प्रणाली भी। इस एम्बेडेड सिस्टम की निम्नलिखित विशेषताएँ होती है ।

1. एंबेडेड सिस्टम पूर्व-प्रोग्राम किए गए कार्यों को निष्पादित करते हैं और इनमें आवश्यकताओं का एक विशेष सेट होता है। ये प्रोग्राम किए गए हार्डवेयर डिवाइस हैं, जो हार्डवेयर चिप पर चलते हैं जो प्रोग्राम करने योग्य हैं।
2. एंबेडेड सिस्टम कंप्यूटर के विपरीत एक विशिष्ट कार्य या विशिष्ट कार्यों का एक सेट करते हैं, जिसका उपयोग विस्तृत कार्यों को करने के लिए किया जाता है।
3. ये हमेशा स्वतंत्र उपकरण नहीं होते हैं। एंबेडेड सिस्टम एक बहुत बड़े उपकरण का छोटे भाग होते हैं, जो एक विशिष्ट कार्य को पूरा करते हैं। उदाहरण के लिए, गिब्सन रोबोट गिटार गिटार के तार को ट्यून करने के लिए एक एम्बेडेड सिस्टम का उपयोग करता है, लेकिन गिटार का प्राथमिक कार्य संगीत बजाता है। कारों में एम्बेडेड सिस्टम होते हैं, जो कई सहायक कार्यों को पूरा करते हैं भले ही मुख्य उद्देश्य परिवहन हो।
4. एम्बेडेड सिस्टम के लिए स्क्रिप्ट किए गए प्रोग्राम को फर्मवेयर कहा जाता है और इसे ROM या फ्लैश मेमोरी में स्टोर किया जाता है।
5. एंबेडेड सिस्टम में न तो कोई यूजर इंटरफेस हो सकता है और न ही अत्यधिक उन्नत ग्राफिकल इंटरफेस हो सकता है। यह मुख्य रूप से डिवाइस के उद्देश्य या उस फ़ंक्शन पर निर्भर करता है, जिसे इसे बाहर ले जाने के लिए डिज़ाइन किया गया है। सरल एम्बेडेड सिस्टम सरल मेन्यु विकल्पों के साथ एलईडी, बटन या एलसीडी डिस्प्ले का उपयोग करते हैं। कॉम्प्लेक्स डिवाइस स्पर्श स्क्रीन को लचीलापन प्रदान करने और एक ही समय में उपयोग कर सकते हैं, स्थान की आवश्यकता को कम कर सकते हैं।

4.14 एम्बेडेड सिस्टम के लक्षण (Characteristics of Embedded Systems)

एंबेडेड सिस्टम की प्रमुख विशेषताएँ निम्नलिखित हैं—

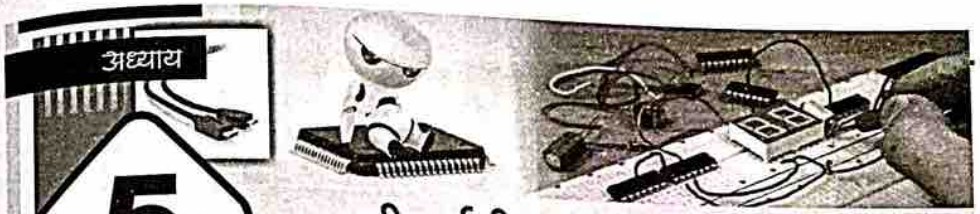
- एंबेडेड सिस्टम के सभी कार्य विशिष्ट हैं। वे अपने जीवनकाल में लगातार एक ही कार्य करते रहते हैं। एक MP3 प्लेयर केवल एक MP3 प्लेयर के रूप में कार्य करेगा।
- एक निश्चित समय सीमा कार्य करने के लिए एंबेडेड सिस्टम बनाए जाते हैं। इसलिए यह पर्याप्त तेजी से प्रदर्शन करता है। कार की ब्रेक प्रणाली, यदि समय सीमा से अधिक है, तो दुर्घटना हो सकती है।
- इनमें न्यूनतम या कोई उपयोगकर्ता इंटरफ़ेस (UI) नहीं है। एक पूर्ण रूप से स्वचालित वॉशिंग मशीन प्रोग्राम सेट होने के बाद स्वयं कार्य करती है और एक बार कार्य समाप्त होने के बाद स्वत: रुक जाती है।
- कुछ एम्बेडेड सिस्टम जैसे—थर्मामीटर एवं जी०पी०एस० ट्रेकिंग डिवाइस बाहरी उत्तेजनाओं पर प्रतिक्रिया करने और तदनुसार प्रतिक्रिया करने के लिए डिज़ाइन किए गए हैं।
- एंबेडेड सिस्टम कुछ दक्षता स्तरों को प्राप्त करने के लिए बनाए गए हैं। ये छोटे आकार के होते हैं, कम शक्ति के साथ कार्य कर सकते हैं और बहुत महंगे भी नहीं हैं।
- एंबेडेड सिस्टम को उपयोगकर्ताओं द्वारा बदला या उन्नत नहीं किया जा सकता है। इसलिए, इनमें विश्वसनीयता और स्थिरता पर उच्च रैंक चाहिए। उपयोगकर्ता बिना किसी कठिनाई का अनुभव किए लंबे समय तक कार्य करते हैं।
- माइक्रोकंट्रोलर या माइक्रोप्रोसेसर का उपयोग एम्बेडेड सिस्टम को बनाने के लिए किया जाता है।
- एंबेडेड सिस्टम में इनपुट और आउटपुट डिवाइस को जोड़ने के लिए कनेक्टेड पेरिफेरल की आवश्यकता होती है।
- एक एम्बेडेड-सिस्टम का हार्डवेयर सुरक्षा और प्रदर्शन के लिए उपयोग किया जाता है। सॉफ्टवेयर का उपयोग सुविधाओं के लिए किया जाता है।



चित्र 4.3 : एम्बेडेड सिस्टम की विशेषताएँ

प्रश्नावली

1. एम्बेडेड सिस्टम के बारे में चर्चा कीजिए।
2. एम्बेडड सिस्टम की संरचना बनाकर व्याख्या कीजिए।
3. एम्बेडडेर सिस्टम की कार्य प्रणाली की संक्षिप्त व्याख्या कीजिए।
4. रीयल टाइम ऑपरेटिंग सिस्टम (Real Time Operating System RTOS) के बारे में व्याख्या कीजिए।
5. एम्बेडेड सिस्टम को प्रभावित करने वाले कारकों की व्याख्या कीजिए।
6. एम्बेडेड सिस्टम के विभिन्न अनुप्रयोगों की विस्तृत व्याख्या कीजिए।
7. एम्बेडेड सिस्टम के अभिलक्षण और गुणों की व्याख्या कीजिए।
8. एम्बेडेड सिस्टम में विभिन्न प्रकार की मेमोरी के बारे में लिखिए।
9. Small scale और Medium scale embedded system की व्याख्या कीजिए।
10. एम्बेडेड सिस्टम के अभिलक्षण लिखिए।
11. एम्बेडेड सिस्टम की चुनौतियों के बारे में लिखिए।



अध्याय 5

पीआईसी माइक्रोकंट्रोलर का परिचय
(Introduction of PIC Microcontroller)

SYLLABUS

Introductin of PIC micro-controller, block diagram, function of each block, introduction of AVR micro-controller, block diagram.

5.1 PIC माइक्रोकंट्रोलर का परिचय (Introduction of PIC Micro-controller)

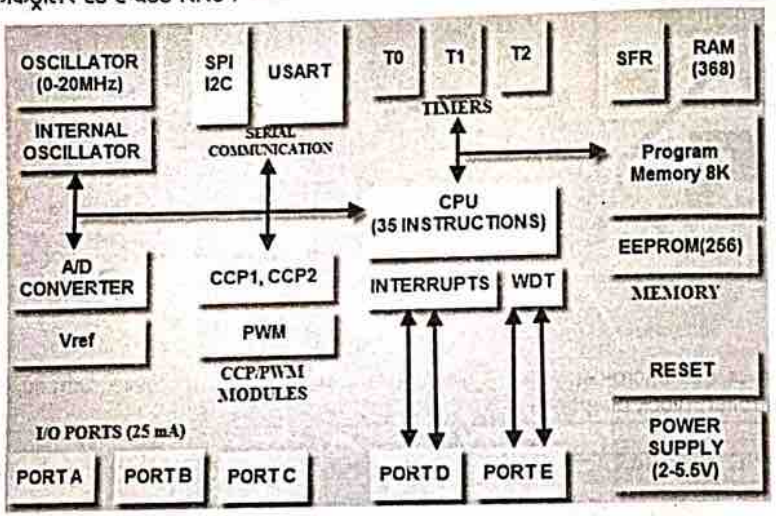
PIC एक पेरिफेरेल इंटरफ़ेस माइक्रोकंट्रोलर (Peripheral Interface Microcontroller) है, जिसको जनरल इंस्ट्रूमेंट माइक्रोकंट्रोलर द्वारा सन् 1993 में विकसित किया गया था। इसको सॉफ्टवेर से नियंत्रित किया जाता है और अलग अलग प्रकार के कार्य करने के लिए प्रोग्राम किया जाता है PIC माइक्रोकंट्रोलर का उपयोग नए-नए अनुप्रयोगों, जैसे—स्मार्ट फोन , ऑडियो डिवाइस, आधुनिक मैडिकल डिवाइस आदि में किया जाता है।

चित्र 5.1 : PIC माइक्रोकंट्रोलर

बाजार में PIC16F84 से PIC16C84 तक कई PIC उपलब्ध हैं। माइक्रोचिप ने हाल ही में विभिन्न प्रकारों के साथ फ्लैश चिप प्रस्तुत किए हैं, जैसे—16F628, 16F877, और 18F452। 16F877 का मूल्य पुराने 16F84 से दोगुना है, लेकिन यह कोड साइज़ से आठ गुना अधिक है, अधिक रैम और बहुत अधिक I / O पिन, एक UART, A/D कनवर्टर और बहुत अधिक विशेषताओं के साथ उपलब्ध है ।

5.2 PIC माइक्रोकंट्रोलर का आर्किटेक्चर (PIC Micro-controllers Architecture)

PIC माइक्रोकंट्रोलर के आर्किटेक्चर में CPU, I/O पोर्ट, मेमोरी ऑर्गनाइजेशन, A/D कन्वर्टर, टाइमर/काउंटर, इंटरप्ट, सीरियल कम्युनिकेशन, ओस्सिलेटर और CCP मॉड्यूल शामिल हैं, जिनके बारे में इस अध्याय में विस्तार से बताया गया है।



चित्र 5.2 : PIC माइक्रोकंट्रोलर का आर्किटेक्चर

5.2.1 सेंट्रल प्रोसेसिंग यूनिट CPU (Central Processing Unit)

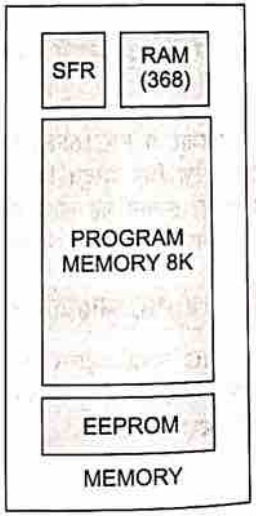
यह अन्य माइक्रोकंट्रोलर CPU से अलग नहीं है PIC माइक्रोकंट्रोलर CPU में ALU, CU, MU और एक्युमुलेटर (Accumulator) आदि होते हैं। अरिथमेटिक लॉजिकल यूनिट (ALU) मुख्य रूप से अंकगणितीय संचालन के लिए और तार्किक निर्णय लेने के लिए उपयोग की जाती है। मेमोरी यूनिट (MU) का उपयोग प्रोसेसिंग के बाद निर्देशों को संग्रहीत करने के लिए किया जाता है। आंतरिक और बाह्य उपकरणों को नियंत्रित करने के लिए कंट्रोल यूनिट (CU) का उपयोग किया जाता है, जो CPU से जुड़े होते हैं और एक्युमुलेटर का उपयोग परिणामों और आगे की प्रक्रिया को संग्रहीत करने के लिए किया जाता है।

5.2.2 मेमोरी आर्गेनाइजेशन (Memory Organization)

PIC माइक्रोकंट्रोलर में मेमोरी मोड्यूल में RAM (Random Access Memory), ROM (Read Only Memory) और STACK होता है।

1. रैंडम एक्सेस मेमोरी Random Access Memory (RAM)—RAM एक अस्थिर मेमोरी है, जिसका उपयोग डाटा को अपने रजिस्टरों में अस्थायी रूप से संग्रहीत करने के लिए किया जाता है। RAM मेमोरी को दो मेमोरी वर्गों में वर्गीकृत किया गया है, और प्रत्येक वर्ग में बहुत सारे रजिस्टर होते हैं। RAM रजिस्टर को दो प्रकारों में वर्गीकृत किया जाता है—Special Function Registers (SFR)) और General Purpose Registers (GPR)।

(i) **जनरल परपज रजिस्टर** (General Purpose Registers (GPR))—जैसा कि नाम का ही अर्थ है इन रजिस्टरों का उपयोग सामान्य प्रयोजन के लिए किया जाता है। उदाहरण के लिए, यदि हम PIC माइक्रोकंट्रोलर का उपयोग करके दो संख्याओं की गुणा करना चाहते हैं। तो हम अन्य रजिस्टरों में संख्याओं को गुणा



चित्र 5.3 : Memory Organization

करने और संग्रहीत करने के लिए रजिस्टरों का उपयोग करते हैं। इसलिए इन रजिस्टरों का कोई विशेष कार्य नहीं है—CPU आसानी से रजिस्टरों में डाटा का उपयोग कर सकता है।

(ii) **स्पेशल फंक्शन रजिस्टर** (Special Function Registers)—इन रजिस्टर्स का उपयोग विशेष प्रयोजनों के लिए किया जाता है। ये रजिस्टर उन्हें सौंपे गए कार्यों के अनुसार कार्य करते हैं। इनका उपयोग सामान्य रजिस्टरों के रूप में नहीं किया जा सकता है। उदाहरण के लिए, आप डाटा को संग्रहीत करने के लिए STATUS रजिस्टर का उपयोग नहीं कर सकते हैं, इन रजिस्टरों का उपयोग प्रोग्राम के संचालन या स्थिति को दिखाने के लिए किया जाता है। इसलिए, उपयोगकर्ता SFR का कार्य नहीं बदल सकता है; फंक्शन मैन्यूफैक्च्युरिंग के समय रिटेलर द्वारा दिया जाता है।

2. रीड ओनली मेमोरी (Read Only Memory (ROM))— ROM एक स्थिर मेमोरी है, जिसका उपयोग डाटा को स्थायी रूप से संग्रहीत करने के लिए किया जाता है। PIC माइक्रोकंट्रोलर आर्किटेक्चर में ROM माइक्रोकंट्रोलर प्रोग्राम के अनुसार इंस्ट्रक्शन या प्रोग्राम को स्टोर करता है। ROM को प्रोग्राम मेमोरी भी कहा जाता है, जिसमें उपयोगकर्ता माइक्रोकंट्रोलर के लिए प्रोग्राम लिखेगा और इसे स्थायी रूप से सुरक्षित करेगा, और अंत में CPU द्वारा प्रोग्राम को निष्पादित (Execute) किया जाता है। माइक्रोकंट्रोलर्स का प्रदर्शन इंस्ट्रक्शन पर निर्भर करता है, जिसे सीपीयू द्वारा एक्सीक्यूट किया जाता है।

3. स्टैक (Stack)—जब कोई व्यवधान (Interrupt) होता है, तो पहले PIC माइक्रोकंट्रोलर को व्यवधान और उपस्थित प्रक्रिया एड्रेस को Execute करना पड़ता है। फिर जो Execute किया जा रहा है वह Stack में संग्रहीत होता है। व्यवधान के निष्पादन (Execution) को पूरा करने के बाद, माइक्रोकंट्रोलर प्रक्रिया को एड्रेस की मदद से कॉल करता है, जो Stack में संग्रहीत होता है और प्रक्रिया को निष्पादित करता है।

3. I/O पोर्ट्स (I/O Ports)

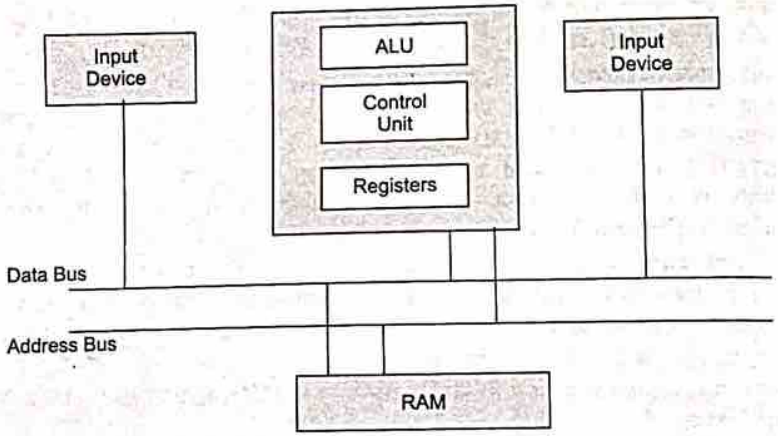
- PIC16 की श्रृंखला में पांच पोर्ट Port A, Port B, Port C, Port D and Port E होते हैं।
- Port A एक 16-bit का पोर्ट होता है, जो TRISA (Tradoc Intelligence Support Activity) रजिस्टर के आधार पर इनपुट या आउटपुट पोर्ट की तरह उपयोग किया जाता है।
- Port B एक 8- bit का पोर्ट है, जिसको इनपुट और आउटपुट पोर्ट की तरह उपयोग किया जाता है।
- Port C एक 8-bit का पोर्ट है और आउटपुट ऑपरेशन का इनपुट TRISC रजिस्टर के स्टेटस से निर्धारित किया जाता है।
- Port D एक 8-bit का पोर्ट है, जो माइक्रोप्रोसेसर BUS से कनेक्शन के लिए स्लेव पोर्ट की तरह कार्य करता है।
- Port E एक 3-bit का पोर्ट है, जो एनालॉग से डिजिटल कनवर्टर तक नियंत्रण संकेतों के अतिरिक्त कार्य करता है।

4. BUS

बस का उपयोग एक पेरिफेरेल से दूसरे में डाटा को स्थानांतरित करने और प्राप्त करने के लिए किया जाता है। इसे दो वर्गों में वर्गीकृत किया जाता है, डाटा बस और एड्रेस बस।

(i) **डाटा बस** (Data Bus)—इसका उपयोग केवल डाटा ट्रांसफर या प्राप्त करने के लिए किया जाता है।

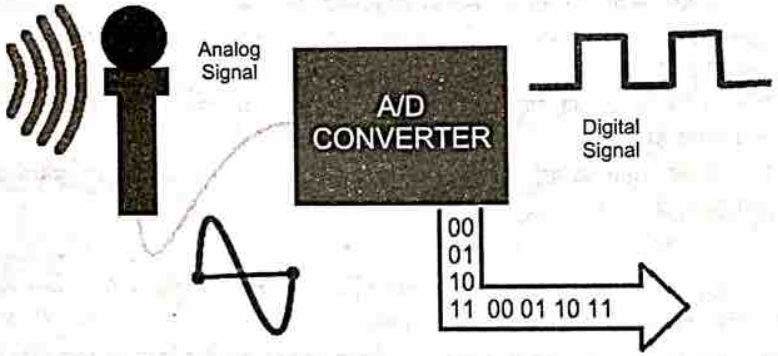
(ii) **एड्रेस बस** (Address Bus)—एड्रेस बस का उपयोग पेरिफेरेलस (Peripherals) से मेमोरी एड्रेस को सीपीयू में प्रसारित करने के लिए किया जाता है। I/O पिन बाहरी पेरिफेरेलस को इंटरफेस करने के लिए उपयोग किया जाता है; UART और USART दोनों सीरियल संचार प्रोटोकॉल हैं, जिनका उपयोग सीरियल उपकरणों, जैसे—GSM, GPS, ब्लूटूथ, IR, आदि के लिए किया जाता है।



चित्र 5.4 : BUS

5. A/D Converters

A/D कनवर्टर का मुख्य उद्देश्य एनालॉग वोल्टेज को डिजिटल वोल्टेज में बदलना है। PIC माइक्रोकंट्रोलर के A/D मॉड्यूल में 28 पिन डिवाइस के लिए 5 इनपुट और 40 पिन डिवाइस के लिए 8 इनपुट होते हैं। A/D कनवर्टर का संचालन ADCON0 और ADCON1 विशेष रजिस्टरों द्वारा नियंत्रित किया जाता है। कनवर्टर के अपर बिट्स (Upper bits) रजिस्टर ADRESH में संग्रहीत किए जाते हैं और कनवर्टर के निचले बिट्स (Lower bits) रजिस्टर ADRESL में संग्रहीत किए जाते हैं। इस ऑपरेशन के लिए एक एनालॉग संदर्भ वोल्टेज के 5V की आवश्यकता होती है।



चित्र 5.5 : AD Converter

6. टाइमर्स / काउंटर्स (Timers/ Counters)

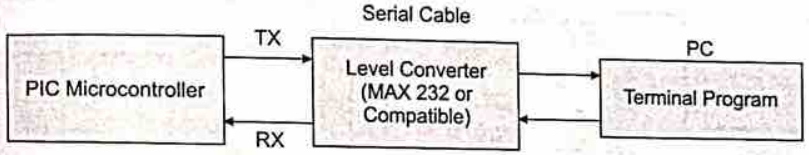
PIC माइक्रोकंट्रोलर में चार टाइमर / काउंटर होते हैं, जिसमें एक 8-बिट का टाइमर और शेष टाइमर में 8 या 16-बिट के मोड का चयन करने का विकल्प होता है। टाइमर का उपयोग सटीकता कार्यों को उत्पन्न करने के लिए किया जाता है, उदाहरण के लिए, दो ऑपरेशनों के बीच विशिष्ट समय में देरी करना।

7. व्यवधान (Interrupts)

PIC माइक्रोकंट्रोलर में 20 आंतरिक Interrupts और तीन बाहरी Interrupts स्रोत होते हैं, जो विभिन्न पेरिफेरेल्स , जैसे—ADC, USART, टाइमर, और इसी तरह से जुड़े होते हैं।

सीरियल कम्युनिकेशन एक संचार चैनल पर क्रमिक रूप से एक समय में डाटा को स्थानांतरित करने की विधि है।

- **USART**—USART का पूरा नाम Universal Synchronous And Asynchronous Receiver and Transmitter है, जो दो प्रोटोकॉल के लिए एक सीरियल कम्युनिकेशन है। यह क्लॉक पल्स के साथ में एकल तार पर बिट को प्रसारित करने और प्राप्त करने के लिए उपयोग किया जाता है। PIC माइक्रोकंट्रोलर में दो पिन TXD और RXD होती हैं। इन पिनों का उपयोग क्रमिक रूप से डाटा को प्रसारित करने और प्राप्त करने के लिए किया जाता है ।
- **SPI Protocol**—SPI शब्द का अर्थ Serial Peripheral Interface है। इस प्रोटोकॉल का उपयोग PIC माइक्रोकंट्रोलर और अन्य पेरिफेरेल्स, जैसे—SD कार्ड, सेंसर और शिफ्ट रजिस्टर के बीच डाटा भेजने के लिए किया जाता है। PIC माइक्रोकंट्रोलर कॉमन क्लॉक पल्स पर दो उपकरणों के बीच तीन तार SPI संचार का समर्थन करते हैं। SPI प्रोटोकॉल की डाटा दर USART से अधिक होती है।
- **I2C Protocol**—I2C शब्द का अर्थ Inter Integrated Circuit है, और यह एक सीरियल प्रोटोकॉल है, जिसका उपयोग EEPROMS, माइक्रोकंट्रोलर, A/D कन्वर्टर्स आदि जैसे कम गति वाले उपकरणों को जोड़ने के लिए किया जाता है। PIC माइक्रोकंट्रोलर दो उपकरणों के लिए दो बायर इंटरफेस या I2C संचार का समर्थन करते हैं, जो मास्टर और स्लेव डिवाइस दोनों के रूप में काम करते हैं।



चित्र 5.6 : Protocol

8. ओस्सिलेटर (Occilator)

ऑस्सिलेटर का उपयोग टाइमिंग जनरेशन के लिए किया जाता है। PIC माइक्रोकंट्रोलर में RC ऑसिलेटर या क्रिस्टल ऑसिलेटर्स जैसे बाहरी ऑसिलेटर होते हैं। जहाँ क्रिस्टल ऑसिलेटर दो ऑसिलेटर पिन के बीच जुड़ा होता है। इसके मोड क्रिस्टल मोड, हाई-स्पीड मोड और लो-पावर मोड होते हैं। RC ऑसिलेटर्स में रेसिस्टर और संधारित्र का मान क्लॉक आवृत्ति निर्धारित करती है और क्लॉक आवृत्ति की सीमा 30 kHz से 4 MHz होती है।

9. CCP मोड्यूल (CCP Module)

CCP मॉड्यूल नाम कैप्चर / कम्पेयर / PWM (Capture/compare/PWM) है, जहाँ यह कैप्चर मोड, तुलना मोड और PWM मोड जैसे तीन मोड में काम करता है।

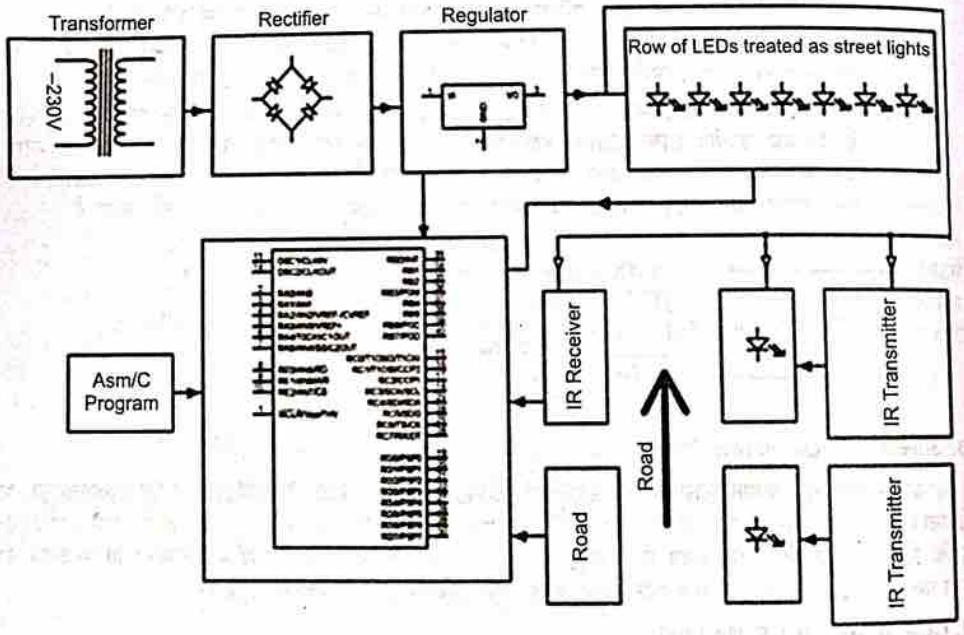
1. **कैप्चर मोड** (Capture Mode)—कैप्चर मोड सिग्नल के आने के समय को पकड़ता है, अन्य शब्दों में, जब सीसीपी पिन हाई होती है, तो यह टाइमर 1 के मान को पकड़ लेता है।
2. **तुलना मोड** (Compare Mode)—तुलना मोड एक Analog compare के रूप में कार्य करता है। जब टाइमर 1 मान एक निश्चित संदर्भ मूल्य तक पहुँचता है, तो यह एक आउटपुट उत्पन्न करता है।
3. **PWM मोड**—PWM मोड एक 10-बिट रिज़ॉल्यूशन और प्रोग्रामेबल ड्यूटी चक्र के साथ पल्स चौड़ाई Modulate आउटपुट प्रदान करता है।

5.3 PIC माइक्रोकंट्रोलर के अनुप्रयोग (Applications of PIc Micro-controller)

PIC माइक्रोकंट्रोलर का उपयोग विभिन्न अनुप्रयोगों में किया जा सकता है, जैसे—पेरिफेरेल्स, ऑडियो सहायक उपकरण, वीडियो गेम, आदि। इस PIC माइक्रोकंट्रोलर के अनुप्रयोग निम्नलिखित प्रोजेक्ट्स में इस प्रकार है—

5.3.1 स्ट्रीट लाइट जो कि वाहन चालन का पता लगाने पर चमकती है (Street Light that Glows on Detecting Vehicle Movement)

इस प्रोजेक्ट का मुख्य उद्देश्य राजमार्गों पर वाहनों की आवाजाही होने पर स्ट्रीट लाइट को स्विच ऑन करना है, और ऊर्जा को संरक्षित करने के लिए ट्रेलिंग लाइट को भी बंद करना है। इस प्रोजेक्ट में, PIC माइक्रोकंट्रोलर को असेम्बली लैंग्वेज (Assembly language) या एम्बेडेड C (Embedded C) का उपयोग करके प्रोग्राम किया जाता है। जिसका सर्किट डायग्राम निचे चित्र में दर्शाया गया है।



चित्र 5.7 : स्ट्रीट लाइट का डायग्राम

इस सर्किट को पॉवर सप्लाई AC पॉवर सप्लाई को स्टेप डाउन, रेक्टिफाई & फ़िल्टर करके दी जाती है। जब राजमार्ग पर कोई वाहन नहीं होगा, तो सभी लाइटें बंद हो जाएँगी, ताकि बिजली का संरक्षण किया जा सके। वाहन की गति को समझने के लिए IR सेंसर सड़क पर रखे जाते हैं। जब राजमार्ग पर वाहन होते हैं, तो IR सेंसर वाहन की गति को तुरंत भांप लेता है, यह सेंसर LED को चालू / बंद करने के लिए PIC माइक्रोकंट्रोलर को कमांड भेजता है। वाहन के सेंसर के पास आने पर LEDs का एक गुच्छा परिचय हो जाएगा और एक बार सेंसर से वाहन गुजर जाने के बाद LED की तीव्रता कम हो जाएगी।

5.4 PIC माइक्रोकंट्रोलर के लाभ (Advantages of PIC Micro-controller)

1. PIC माइक्रोकंट्रोलर सुसंगत (Consistent) हैं और PIC प्रतिशत की दोषपूर्णता बहुत कम है। RISC आर्किटेक्चर का उपयोग करने के कारण PIC माइक्रोकंट्रोलर का प्रदर्शन (Performance) बहुत तेज (Fast) होता है।
2. अन्य माइक्रोकंट्रोलर्स की तुलना में इनमे बिजली की खपत बहुत कम होती है और प्रोग्रामिंग भी बहुत आसान होती है।
3. किसी भी अतिरिक्त सर्किट के बिना एनालॉग डिवाइस का इंटरफेस आसान होता है।

5.5 PIC माइक्रोकंट्रोलर के नुकसान (Disadvantages of PIC Micro-controller)

1. RISC Architecture (35 instructions) का उपयोग करने के कारण प्रोग्राम की लंबाई अधिक होती है।
2. संचायक उपस्थित है और प्रोग्राम मेमोरी सुलभ नहीं होता है।

5.6 AVR माइक्रोकंट्रोलर का परिचय (Introduction of AVR Micro-controller)

AVR Atmel द्वारा सन् 1996 के बाद से विकसित किए गए। यह माइक्रोकंट्रोलर्स का एक परिवार है, जिसे सन् 2016 में माइक्रोचिप टेक्नोलॉजी द्वारा अधिगृहीत किया गया था। AVR पहला माइक्रोकंट्रोलर था जिसमे प्रग्राम स्टोरेज के लिए ऑन चिपफ्लैश मेमोरी का उपयोग किया गया जबकि अन्य माइक्रोकंट्रोलर में एक बार प्रोग्रामेबल ROM, EPROM, या EEPROM का उपयोग होता था। AVR माइक्रोकंट्रोलर के कई एम्बेडेड सिस्टम के अनुप्रयोग हैं। वे विशेष रूप से हॉबीस्ट और शैक्षिक एम्बेडेड अनुप्रयोगों में मुख्य हैं,ओपन हार्डवेयर डवलपमेंट बोर्डों के कई Arduino लाइन में उनके शामिल किए जाने से लोकप्रिय हैं।

चित्र 5.8 : AVR माइक्रोकंट्रोलर

हम पर्सनल कंप्यूटर (PC) के साथ तुलना करके माइक्रोकंट्रोलर को समझ सकते हैं, जिसमें एक मदरबोर्ड होता है। उस मदरबोर्ड में एक माइक्रोप्रोसेसर (AMD, Intel chip) का उपयोग किया जाता है, जो सीरियल पोर्ट, डिस्प्ले इंटरफेस और डिस्क ड्राईवर जैसे सिस्टम में इंटरफेस के लिए इंटेलिजेंस, EEPROM और RAM मेमोरी प्रदान करता है। एक माइक्रोकंट्रोलर में एक चिप में निर्मित सभी या अधिकांश विशेषताएँ होती हैं, इसलिए इसे मदरबोर्ड और किसी अन्य कॉम्पोनेन्ट की आवश्यकता नहीं होती है ।

AVR माइक्रोकंट्रोलर अलग-अलग कॉन्फ़िगरेशन में आता है, कुछ को सरफेस माउंटिंग (Surface mounting) का उपयोग करके और कुछ होल माउंटिंग (Hole mounting) का उपयोग करके डिज़ाइन किया जाता है। यह 8-पिन से 100-पिन के साथ उपलब्ध होता है, कोई भी माइक्रोकंट्रोलर 64-पिन या इससे अधिक के साथ केवल सरफेस माउंट ही होता है।

अधिक प्रयोग किए जाने वाले कुछ AVR माइक्रोकंट्रोलर इस प्रकार है—

- ATmega8 माइक्रोकंट्रोलर
- ATmega16 माइक्रोकंट्रोलर
- ATmega32 माइक्रोकंट्रोलर
- ATmega328 माइक्रोकंट्रोलर

5.7 AVR माइक्रोकंट्रोलर के प्रकार (Types of AVR Micro-controller)

AVR माइक्रोकंट्रोलर मुख्यतः तीन प्रकार के होते हैं—

1. Tiny AVR—इस माइक्रोकंट्रोलर में मेमोरी कम होती है और यह आकार में छोटा है इसका उपयोग कुछ आसान अनुप्रयोगों में ही किया जाता है।

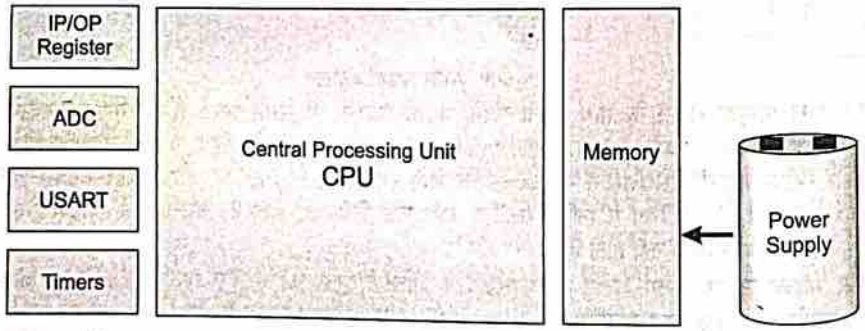
2. Mega AVR—यह माइक्रोकंट्रोलर सबसे अधिक प्रसिद्ध है इसमें 256KB तक मेमोरी होती है और अधिक मात्रा में इनबिल्ट पेरिफेरेल्स होते हैं, जो कि सभी प्रकार के अनुप्रयोगों के लिए महत्त्वपूर्ण होते है।

3. Xmega AVR—इस माइक्रोकंट्रोलर का उपयोग कंपाउंड अनुप्रयोगों के लिए व्यावसायिक रूप से किया जाता है, जिनमे बड़े प्रोग्राम मेमोरी और उच्च गति की भी आवश्यकता होती है।

Series Name	Pins	Flash Memory	Special Feature
Tiny AVR	6-32	0.5-8 KB	Small in size
Mega AVR	6-32	4-256KB	Extended peripherals
Xmega AVR	44-100	16-384KB	DMA, Event System included

5.8 AVR माइक्रोकंट्रोलर आर्किटेक्चर (AVR Micro-controller Architecture)

AVR के आर्किटेक्चर को नीचे चित्र में दिखाया गया है, यह 'हार्वर्ड आर्किटेक्चर' का उपयोग करता है और इस प्रकार इसमें डाटा और प्रोग्राम के लिए अलग-अलग बस (BUS) और मेमोरी हैं। निर्देश प्रोग्राम मेमोरी में सिंगल लेवल पाइपलाइनिंग के साथ Performed होते हैं। जबकि एक इंस्ट्रक्शन को प्राप्त किया जा रहा है इसके साथ ही बाद के निर्देश को प्रोग्राम मेमोरी से प्री-फेच्ड (pre-fetched) किया जाता है। यह विचार इंस्ट्रक्शन को प्रत्येक CLK साइकिल में निष्पादित (Performed) करने की अनुमति देता है और AVR लगभग 1 MIPS/MHz पर चलता है।



चित्र 5.9 Block Diagram Showing Architecture of AVR Micro-controller

1. CPU

AVR माइक्रोकंट्रोलर का CPU कम्प्यूटर के समान और सरल होता है। सीपीयू का मुख्य उद्देश्य सही प्रोग्राम परफॉर्मेंस (Correct program performance) की पुष्टि करना है। इसलिए, CPU को गणना करना (Perform calculations), मेमोरी (Memory), कण्ट्रोल पेरिफेरेल्स (Control peripherals) को संभालने और बाधित करने (Interrupts) में सक्षम होना चाहिए। Atmel के 8-बिट और 32-बिट AVR के CPU 'हार्वर्ड आर्किटेक्चर' पर आधारित हैं, इस प्रकार प्रत्येक IC में दो बस होती हैं, जिनमें एक इंस्ट्रक्शन बस और एक डाटा बस होती है। CPU इंस्ट्रक्शन बस में निष्पादन योग्य निर्देशों (Executable instructions) को पढ़ता है, जबकि डाटा बस का कार्य, सम्बंधित डाटा को पढ़ना या लिखना होता है। AVR के CPU कोर में ALU, सामान्य प्रयोजन रजिस्टर (General Purpose Registers), प्रोग्राम काउंटर (Program Counter), निर्देश रजिस्टर (Instruction Register), निर्देश डिकोडर (Instruction Decoder), स्थिति रजिस्टर (Status Register) और स्टैक पॉइंटर (Stack Pointer) होता हैं।

2. फ्लैश प्रोग्राम मेमोरी (Flash Program Memory)

AVR माइक्रोकंट्रोलर का प्रोग्राम नॉन-वोलेटाइल प्रोग्रामेबल फ्लैश प्रोग्राम मेमोरी (Non-volatile programmable Flash program memory) में संग्रहीत किया जाता है, जो आपके SD कार्ड या Mp3 प्लेयर में फ्लैश स्टोरेज के समान है। फ्लैश प्रोग्राम मेमोरी को दो इकाइयों में विभाजित किया गया है। पहली इकाई एप्लीकेशन फ्लैश सेक्शन है। यह वह स्थान है जहाँ AVR का प्रोग्राम संग्रहीत किया जाता है। दूसरे खंड को बूट फ्लैश अनुभाग के रूप में नामित किया गया है और पॉवर अप होने पर डिवाइस को संचालित करने पर सीधे प्रदर्शन करने के लिए फिक्स्ड किया जा सकता है। ध्यान देने योग्य एक महत्वपूर्ण तथ्य यह है, कि माइक्रोकंट्रोलर्स फ्लैश प्रोग्राम मेमोरी में कम-से-कम 10,000 चक्रों को लिखने / मिटाने (10,000 writes/eras e cycles) का Resolution होता है।

3. Static Random Access Memory (SRAM)

AVR माइक्रोकंट्रोलर का SRAM (Static Random Access Memory) कंप्यूटर रैम की तरह ही होता है। इसमें रजिस्टर का उपयोग गणना को निष्पादित (Execute) करने के लिए किया जाता है, SRAM का उपयोग रनटाइम के माध्यम से डाटा की आपूर्ति करने के लिए किया जाता है। यह वोलेटाइल मेमोरी 8-बिट रजिस्टरों में पूर्वस्थापित होती है।

4. Electronic Enable Programmable Read Only Memory (EEPROM)

EEPROM का अर्थ इलेक्ट्रॉनिक इरेज़ेबल रीड-ओनली मेमोरी होता है, यह एक नॉन-वोलेटाइल मेमोरी होती है, लेकिन आप इससे कोई प्रोग्राम नहीं चला सकते हैं, लेकिन इसका उपयोग लंबे समय तक स्टोरेज के रूप में किया जाता है। यह डाटा जैसे डिवाइस पैरामीटर और रनटाइम पर सिस्टम के कॉन्फ़िगरेशन को स्टोर करने का स्थान है।

5. डिजिटल I/O मोड्यूल्स (Digital I/O Modules)

डिजिटल I/O मॉड्यूल AVR माइक्रोकंट्रोलर और बाह्य युक्तियों के साथ डिजिटल संचार या लॉजिक का संचार करते हैं। संचार संकेत TTL/CMOS लॉजिक के होते हैं।

6. एनालॉग I/O मोड्यूल्स (Analog I/O Modules)

एनालॉग I/O मॉड्यूल का उपयोग इनपुट या आउटपुट एनालॉग सूचना से या बाह्य युक्तियों के लिए किया जाता है। इन मॉड्यूल्स में एनालॉग कंप्रेटरस और एनालॉग-टू-डिजिटल कन्वर्टर्स (ADC) शामिल होते हैं।

7. इंटरप्ट यूनिट (Interrupt Unit)

इंटरप्ट माइक्रोकंट्रोलर को एप्लिकेशन प्रोग्राम परफॉर्म करने के दौरान बैकग्राउंड में विशेष घटनाओं की निगरानी करने में सक्षम करता है और यदि यूनिक प्रोग्राम को रोकना आवश्यक हो, तो घटना पर प्रतिक्रिया देता है। ये सब इंटरप्ट यूनिट द्वारा सिंक्रोनाइज़ किया गया है।

8. टाइमर (Timer)

अधिकांश AVR माइक्रोकंट्रोलर में कम-से-कम एक टाइमर या काउंटर मॉड्यूल होता है, जो कि माइक्रोकंट्रोलर में समय या गिनती संचालन को प्राप्त करने के लिए उपयोग किया जाता है। इनमें टाइम स्टम्पिंग, घटनाओं को गिनना (Counting events), अंतराल को मापना (Measuring intervals) आदि हैं।

9. वाचडॉग (Watchdog)

सभी AVR माइक्रोकंट्रोलर्स में एक आंतरिक वॉचडॉग टाइमर होता है। इसमें बहुत सीमित उपयोगी विशेषताएँ हैं जैसे अलग-अलग 128kHz CLK स्रोत, माइक्रोकंट्रोलर को रीसेट करने और इंटरप्ट को उत्पन करना।

10. USART / SPI / I2C

USART/ SPI/I2C जैसी इकाइयों का उपयोग बाह्य युक्तियों के साथ सीरियल संचार के लिए किया जाता है। एक उदाहरण USART पेरिफेरेल है, जो RS232 मानक का उपयोग करता है।

5.9 AVR माइक्रोकंट्रोलर्स के प्रकार (Types of Avr Micro-controllers)

AVR माइक्रोकंट्रोलर चार श्रेणियों में उपलब्ध हैं—

1. क्लासिक AVR (AT90S xxxx)—ये मुख्य AVR चिप्स हैं, जिन्हें नए AVR Chip द्वारा बदल दिया गया है। इन क्लासिक AVR को नए डिजाइनों के लिए अनुशंसित (Recommended) नहीं किया गया है। **उदहारण**—AT90S2313, AT90S2323, AT90S4433 आदि।

2. टिनी AVR (ATtiny xx)—इनमें मेमोरी कम होती है, आकार कम होता है और आसान अनुप्रयोगों में प्रयोग किये जाते है। **उदहारण**— ATtiny13, ATtiny25, ATtiny44 आदि।

3. Mega AVR (ATmega xxxx)—ये सबसे लोकप्रिय हैं, जिनमें अच्छी मात्रा में मेमोरी (256 KB तक), अधिक संख्या में इनबिल्ट पेरिफेरल्स और ये मध्यम से जटिल अनुप्रयोगों के लिए उपयुक्त होते हैं। **उदहारण**—ATmega8, ATmega16, ATmega128etc आदि।

4. Xmega AVR (ATmega xx)—इसको व्यावसायिक रूप से जटिल अनुप्रयोगों के लिए उपयोग किया जाता है, जिसमें बड़े प्रोग्राम मेमोरी और उच्च गति की आवश्यकता होती है। **उदहारण**—ATxmegaA4, ATxmegaA3B, ATxmegaA1 आदि।

5.10 AVR माइक्रोकंट्रोलर की विशेषताएँ (Features of AVR Micro-controller)

AVR माइक्रोकंट्रोलर की कुछ मुख्य विशेषताएँ निम्नलिखित हैं जिन्हें विवरण के साथ वर्णित किया गया है।

1. यह मॉड्यूल कई कार्य करता है, जिसमें दो-दिशात्मक इनपुट और आउटपुट पोर्ट होते हैं, जिन्हें पुल-अप प्रतिरोध के साथ कॉन्फ़िगर किया जा सकता है।
2. इनमे कई आंतरिक ऑसिलेटर होते हैं, जिसमें RC ऑसिलेटर्स भी शामिल हैं, जो इस बोर्ड पर लगे होते हैं।
3. इस बोर्ड पर दो 50 Kb स्टोरेज स्पेस की फ्लैश मेमोरी होती है।
4. चार-किलो बाइट के स्थान वाले EEPROM को इस बोर्ड पर असेंबल्ड किया जाता है।
5. 16 KB की Static RAM इस बोर्ड पर लगी होती है।
6. इस बोर्ड पर 8-Bit और 16-Bit निकास के दो टाइमर होते हैं।
7. इस बोर्ड पर पल्स विड्थ मॉड्यूलेशन आउटपुट होता है।
8. इस मॉड्यूल पर एक एनालॉग कोम्परेटर भी होता है।

9. इस बोर्ड पर सोलह चैनल मल्टीप्लेक्स के साथ डिजिटल कनवर्टर में 12 Bit एनालॉग होते हैं।
10. इस बोर्ड पर 12 Bit का एनालॉग से डिजिटल कनवर्टर होता है।
11. इसमें संचार के लिए I2C और TWI इंटरफेसिंग का उपयोग होता है।
12. इसका ऑपरेटिंग वोल्टेज 1.8 वोल्ट होता है।

प्रश्नावली

1. PIC माइक्रोकंट्रोलर क्या होते हैं?
2. PIC माइक्रोकंट्रोलर का ब्लॉक डायग्राम बनाते हुए इसकी व्याख्या कीजिए।
3. PIC माइक्रोकंट्रोलर का ब्लॉक डाइग्राम बनाकर विभिन्न ब्लॉक के कार्यों का उल्लेख कीजिए।
4. AVR माइक्रोकंट्रोलर क्या होते हैं?
5. PIC माइक्रोकंट्रोलर और AVR माइक्रोकंट्रोलर के गुण व दोषों की व्याख्या कीजिए।
6. AVR माइक्रोकंट्रोलर का ब्लॉक डायग्राम बनाकर प्रत्येक ब्लॉक के कार्यों का उल्लेख कीजिए।

❑

माइक्रोकंट्रोलर के प्रोग्रामिंग कॉन्सेप्टस
(Programming Concepts of Microcontroller)

SYLLABUS

Introduction, programming concepts of microcontrollers basic introduction of Software used in micro-controllers, how to transfer C or ASM code in micro-controllers.

6.1 परिचय (Introduction)

माइक्रोकंट्रोलर प्रोग्रामिंग के लिए हमें निम्नलिखित मूलभूत शब्दावली से परिचित होना चाहिए। माइक्रोकंट्रोलर प्रोग्रामिंग में आरम्भ के रूप में, यह निम्नलिखित सभी शब्दावली को समझने के लिए बिल्कुल पर्याप्त है।

Program

प्रोग्रामिंग भाषा में लिखे गए कम्प्यूटर के निर्देशों का सेट जो एक एल्गोरिथ्म को लागू करता है। अंत में असेम्बली निर्देशों में एकत्रित हुए जिन्हें 0's & 1's के रूप में कोडित किया गया और मेमोरी में फ्लैश किया जाता है।

Paging

पृष्ठ मेमोरी का एक लॉजिकल ब्लॉक होता है। एक पृष्ठांकित मेमोरी सिस्टम में एक विशिष्ट मेमोरी की लोकेशन को दर्शाने के लिए एक पृष्ठ एड्रेस और एक विस्थापन एड्रेस का उपयोग किया जाता है।

Bank

मेमोरी की एक लॉजिकल इकाई, जो हार्डवेयर पर निर्भर होती है। एक बैंक का आकार आगे एक कॉलम और एक पंक्ति में बिट्स की संख्या से निर्धारित होता है, एक बैंक में चिप की संख्या से गुणा किया जाता है।

Pointer

मेमोरी में एक विशिष्ट ऑब्जेक्ट का एक पता जो उस ऑब्जेक्ट को दर्शाने के लिए उपयोग किया जाता है।

Stack

हार्डवेयर स्टैक मेमोरी का एक भाग है, जिसका उपयोग अस्थायी डाटा या रजिस्टरों के मूल्यों को संग्रहीत करने के लिए किया जाता है। एक सॉफ्टवेयर स्टैक एक अंतिम-इन-प्रथम आउट (LIFO) डाटा की संरचना का होता है, जिसमें ऐसी जानकारी होती है, जिसे सहेजा (Pushed) और पुनर्स्थापित (Restored) किया जाता है ।

Stack Pointer

एक रजिस्टर जिसमें हार्डवेयर स्टैक के शीर्ष का पता होता है।

Program Counter

इसे संक्षिप्त में PC के रूप में जाना जाता है। यह एक रजिस्टर है, जिसे निष्पादित किए जाने वाले अगले निर्देश का पता होता है। प्रत्येक निर्देश प्राप्त होने के बाद प्रोग्राम काउंटर को बढ़ाया जाता है (It's incremented once/instruction cycle)।

Interrupts

इंटरप्ट एक ऐसी घटना है, जो मुख्य प्रोग्राम के निष्पादन को निलंबित कर देती है, जबकि दूसरे प्रोग्राम द्वारा सेवित (नियंत्रित) होती है। इंटरप्ट से बाहरी घटनाओं में सिस्टम की प्रतिक्रिया की गति में काफी वृद्धि होती है।

Interrupt Vector

यह वह स्थान है, जहाँ से प्रोग्राम का निष्पादन जारी होता है। इंटरप्ट वेक्टर वाले स्थान को आमतौर पर नियमित कार्यक्रम निष्पादन के दौरान पार किया जाता है।

Clock

यह एक माइक्रोकंट्रोलर का धड़कता हुआ हृदय होता है। यह एक फिक्स्ड-फ्रीक्वेंसी सिग्नल है, जो सीपीयू ऑपरेशन और घटनाओं को ट्रिगर या सिंक्रनाइज़ करता है। एक क्लॉक में एक आवृत्ति होती है, जो मेगाहर्ट्ज में दोलन की अपनी दर का वर्णन करती है। क्लॉक स्रोत एक बाहरी आरसी नेटवर्क, क्रिस्टल ऑसिलेटर, रेसोनेटर या एक अंतर्निर्मित आंतरिक घड़ी स्रोत भी हो सकता है। दोलन की आवृत्ति को आमतौर पर FOSC कहा जाता है।

Machine Cycle

मशीन चक्र वह समय है, जो सिस्टम की क्लॉक को स्वयं दोहराता है। एक माइक्रोकंट्रोलर के लिए, जिसके क्लॉक इनपुट पिंन पर 4 MHz का क्रिस्टल OSC लगा होता है, मशीन का चक्र 1/4000000 = 250 नैनोसेकंड होगा।

Instruction Cycle

निर्देश चक्र वह समय है, जो किसी एकल निर्देश के निष्पादन को पूरा करने के लिए एक माइक्रोकंट्रोलर लेता है। इसमें MCU के उपयोग के लिए 4 क्लॉक चक्र लेने वाले परिणाम को प्राप्त करना, डिकोड करना, निष्पादन और सहेजना शामिल है। 1 निर्देश चक्र = 4 मशीन चक्र।

We can also say that a CPU that is running @ 4 MHz executes FOSC/4 instructions per second = 1 MIPS (Million Instructions Per Second).

हम इस अभ्यास के दौरान MCUs प्रोग्रामिंग कर रहे हैं, जब आप PC पर चलने वाले एप्लिकेशन को बनाने के लिए अपने PC पर कोडिंग करते हैं, तो इसे 'Development' प्रक्रिया कहा जाता है। हालाँकि, हमेशा ऐसा नहीं होता है। कई स्थितियों में आप एक विशिष्ट मशीन (जैसे PC) पर कोड लिख रहे होंगे, जो अंततः एक मशीन कोड के लिए संकलित हो जाएगा जो किसी अन्य मशीन (जैसे एंड्रॉइड, आईओएस, एमसीयू, आदि) पर चलता है। इसे हम 'क्रॉस-डेवलपमेंट' कहते हैं, और यह एम्बेडेड फर्मवेयर प्रोग्रामिंग के लिए होता है। क्रॉस-डेवलपमेंट के लिए उपयोग किए जाने वाले कंपाइलर को अक्सर क्रॉस-कंपाइलर कहा जाता है।

एम्बेडेड सॉफ्टवेयर विकसित करने व उपयोग करने के लिए वास्तव में कई प्रोग्रामिंग भाषाएँ उपलब्ध हैं। यद्यपि, C-प्रोग्रामिंग भाषा इस उद्योग में मानक है। यह रजिस्टर स्तर पर निम्न-स्तरीय हार्डवेयर के साथ बातचीत करने के लिए अभी भी सबसे कुशल तरीका माना गया है। अधिकांश उन्नत उच्च-गुणवत्ता वाले सॉफ्टवेयर C भाषा में विकसित किए गए हैं। हम कुछ बिंदुओं पर असेंबली (Assembly) का उल्लेख कर सकते हैं। यह लंबे समय से ज्ञात है, कि आप असेंबली भाषा के साथ अपने सिस्टम में अधिक अनुकूलन कर सकते हैं और यह सच है, लेकिन केवल इस चरण में सीखने के उद्देश्यों के लिए इसका उपयोग करते हैं। अन्यथा, आप लाभदायक कौशल प्राप्त किए बिना बहुत अधिक समय बर्बाद कर रहे हैं। जैसा कि प्रत्येक प्रोसेसर के अपने निर्देश होते हैं, फलस्वरूप इसकी अपनी असेंबली कमांड होती है, इसलिए यह केवल एक शैक्षिक अभ्यास होगा।

जिस प्रोग्रामिंग भाषा में हम PIC MCUs के लिए फर्मवेयर लिखते हैं, उसे C-भाषा, मानक ANSI-C कहते हैं। जो पिछले कुछ दशकों के दौरान एम्बेडेड सॉफ्टवेयर विकास के लिए सबसे कुशल विकल्प रहा है। और आज भी फर्मवेयर लिखने के लिए सबसे उपयुक्त साबित होता है। यह पाठ्यक्रम सी-लैंग्वेज में मुख्य अवधारणाओं के साथ आपकी परिचित की धारणा पर बनाया गया है। यद्यपि, सभी कोड सूचियों को नौसिखिया के अनुकूल लिखा जाता है

जबकि समग्र दक्षता स्तर को यथासंभव अधिक बनाए रखता है। लगभग सभी औद्योगिक मशीनों, सैन्य सेनानियों, रॉकेट, अंतरिक्ष यान, संकलक और यहाँ तक कि अन्य प्रोग्रामिंग भाषाएँ सभी C में बनाई गई हैं। इस तथ्य को ध्यान में रखते हुए, आपको उस व्यक्ति की इस भाषा में अच्छा होने के प्रयास और समय की सराहना करनी चाहिए।

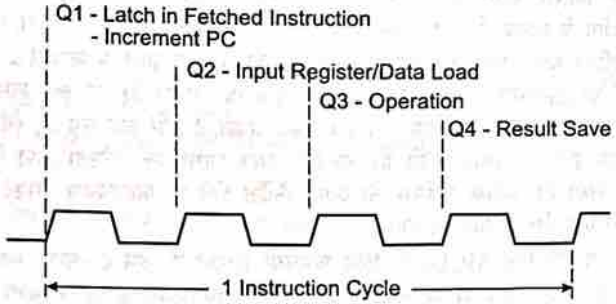
8-बिट MCUs के लिए हम जिस C-Compiler का उपयोग कर रहे हैं, उसे माइक्रोचिप से XC8 कहते हैं। आप इसे अपने OS के लिए उपयुक्त संस्करण डाउनलोड कर सकते हैं। कंपाइलर को सेट-अप करना भी एक सीधी प्रक्रिया है, जिसे आप आसानी से लगा सकते हैं। या आप MPLAB IDE + XC8 कम्पाइलर दोनों को स्थापित करने के लिए अगले ट्यूटोरियल में चरणों का पालन कर सकते हैं।

Integrated Development Environment (IDE) एक सॉफ्टवेयर प्लेटफॉर्म है, जिस पर हम अपने प्रोजेक्ट्स को विकसित कर रहे हैं। यह एक कंप्यूटर सॉफ्टवेयर है, जो उपकरणों के एक समूह को इकट्ठा करता है, जो कम महत्वपूर्ण कार्यों के ओवरहेड को समाप्त करता है। जो डेवलपर्स को पूरे सिस्टम पर अपना ध्यान केंद्रित करने और उनकी प्रमुख समस्याओं को ध्यान में रखने में मदद करता है, जिन्हें अधिक जॉच की आवश्यकता होती है। IDEs के अन्य विकल्प कुछ Pirated (Cracked) सॉफ्टवेयर या ड्रैग-ड्रॉप Arduino हैं।

अब, हम इस बारे में चर्चा करेंगे कि तकनीकी संदर्भ में एक विशिष्ट माइक्रोकंट्रोलर पर एक एम्बेडेड प्रोग्राम कैसे चलता है। सबसे पहले, प्रोग्राम को माइक्रोकंट्रोलर के प्रोग्राम मेमोरी (ROM) में लोड किया जाएगा। C में कोड लिखने के बाद, कंपाइलर और असेंबलर एक *.hex* फाइल बनाएगा, उसके बाद माइक्रोकंट्रोलर चिप का (फ्लैश) जलाना चाहिए। कार्यक्रम के निर्देश 0 और 1 के स्पष्ट रूप से एक समूह हैं। इसके बाद, चिप को किसी भी इलैक्ट्रॉनिक उपकरण के रूप में संचालित किया जासकता है। विशिष्ट माइक्रोकंट्रोलर 3.3 या 5 V DC पर चलते हैं। हमें क्लॉक की इनपुट के साथ चिप प्रदान करने के लिए क्रिस्टल ऑसिलेटर से भी जुड़ना चाहिए।

अब, माइक्रोकंट्रोलर मेमोरी में संग्रहीत निर्देशों को क्रमिक रूप से निष्पादित करना शुरू करेगा। प्रत्येक निर्देश निष्पादन में 4 क्लॉक चक्र होते हैं जो निम्नानुसार काम करते हैं—

1. सीपीयू द्वारा प्रोग्राम मेमोरी से एक निर्देश प्राप्त करता है और प्रोग्राम काउंटर PC प्रोग्राम मेमोरी में अगले निर्देश पर इंगित करने के लिए बढ़ जाता है।
2. Fetched Instruction को दूसरे क्लॉक चक्र में डिकोड किया जाता है।
3. निर्देश का संचालन किया जाता है, यह ADD, SUB, MOV, DIV या जो भी हो सकता है। निष्पादन तीसरे क्लॉक चक्र के दौरान होता है।
4. अंत में, परिणाम चौथे चक्र में कार्य रजिस्टर में सहेजा जाता है। जो पहले प्रोग्राम इंस्ट्रक्शन को निष्पादित करने का अंतिम चरण था।
5. अब, PC 2 निर्देश की ओर संकेत करता है जैसा कि आपने चरण 1 में देखा है और जब तक पॉवर बंद नहीं हो जाती तब तक सब कुछ बार-बार दोहराया जाता है।



चित्र 6.1 : प्रोग्राम काउन्टर

जो स्पष्ट रूप से बताता है, कि क्यों एक मशीन चक्र = 1/4 अनुदेश चक्र या दूसरे शब्दों में, MCU को Fosc/4 MIPS करने के लिए कहा जाता है।

कंप्यूटर सिमुलेशन सॉफ्टवेयर का उपयोग कई एम्बेडेड सिस्टम प्लेटफॉर्म; जैसे—ARM, AVR, PIC, आदि के साथ आपके सीखने के अनुभव में काफी सुधार कर सकता है। यद्यपि, एक विशिष्ट सिमुलेशन सॉफ्टवेयर, उदाहरण—Proteus केवल आपको लॉजिकल (कोड) त्रुटियाँ पकड़ाएगा! यद्यपि यह अभी भी एक शक्तिशाली उपकरण है, जो आपको वास्तविक हार्डवेयर पर फ्लिकरिंग और परीक्षण फ़र्मवेयर से बचने में सहायता करता है, जब तक कि हम इस फ़र्मवेयर को आसानी से साबित नहीं कर सकते हैं कि यह सही ढंग से नहीं चलता है या अपेक्षित परिणाम नहीं देता है। केवल कुछ क्लिक और ड्रैग-ड्रॉप के साथ, आपके पास आपके योजनाबद्ध तरीके पूरी तरह से जुड़े और कुछ परीक्षण के लिए तैयार होते हैं।

एम्बेडेड सिस्टम अभ्यास में आपको विभिन्न हार्डवेयर भागों और विशेष रूप से माइक्रोकंट्रोलर्स के साथ कार्य करना चाहिए, जो निम्नलिखित मुद्दों का परिचय देते हैं। जेनेरिक तरीके से माइक्रोकंट्रोलर्स की प्रोग्रामिंग में उन्हें ब्रेडबोर्ड से निकालना और प्रोग्रामर परिपथ पर माउंट करना शामिल है। जब फर्मवेयर को MCU में फ्लैश किया जाता है, तो आप इसे ब्रेडबोर्ड परीक्षण वातावरण में वापस ले जाते हैं। जैसा कि आपने देखा होगा, कि फर्मवेयर को अपडेट करने में ब्रेडबोर्ड से MCU को हटाना और फिर से जोड़ना होता है। जो संभवतः आपके MCU को हानि पहुँचाएगा और परिणामस्वरूप कुछ मुड़े हुए और टूटे हुए पिन बनेंगे। MCU 3.3 V या 5 V पर संचालित होता है, जिसके लिए आपको आवश्यक है, कि आप अपनी स्वयं की बिजली आपूर्ति और वोल्टेज नियमन ऑनबोर्ड करें ताकि ऑपरेटिंग पॉवर आवश्यक स्पेक्स की गारंटी दे सकें।

उपरोक्त कारणों से हम एक निश्चित ऑन-बोर्ड विकास कनेक्शन से चिपके रहेंगे, जिसका अर्थ है, कि आपको केवल एक बार नीचे मूल सर्किट का निर्माण करना होगा। तब आप अपनी परियोजनाओं को आसानी से बोर्ड पर परख सकेंगे। इस कनेक्शन की निम्नलिखित विशेषताएँ हैं—

- ऑन-बोर्ड ICSP (इन-सर्किट सीरियल प्रोग्रामर) पोर्ट। जिसका अर्थ है, कि आप आसानी से ब्रेडबोर्ड से हटाए बिना चिप को फ्लैश कर सकते हैं।
- चिप को पॉवर देने के लिए विनियमित 5 V विद्युत आपूर्ति कनेक्शन एलईडी संकेतक के साथ होता है।
- रीसेट पिन को ऊपर-नीचे किया जाता है और एक पुश बटन पर हुक किया जाता है।
- ऑसिलेटर इनपुट पिन क्रिस्टल ऑसिलेटर से जुड़े होते हैं।

6.2 माइक्रोकंट्रोलर प्रोग्राम में उपयोग किए जाने वाले उपकरण (Instrument That are Used in Micro-controller Program)

मुख्य रूप से जब हम विभिन्न घटकों को मोटर्स, एलसीडी, एलईडी से कनेक्ट करके एक सर्किट बनाते हैं और बिजली की आपूर्ति देकर जो उस सर्किट द्वारा उपयोग किया जाता है। उस सर्किट के साथ प्रोग्राम किए जाने पर माइक्रोकंट्रोलर क्या करता है? माइक्रोकंट्रोलर के एक प्रोग्राम को समझते हैं, जो असेंबली लेवल लैंग्वेज या C लैंग्वेज में लिखा जाता है, जिसे मशीन लेवल लैंग्वेज में संकलित करना होता है, जिसे बाइनरी लैंग्वेज (यानी zeros & ones) के रूप में जाना जाता है। जिस फाइल को प्रोग्राम किया गया है, वह कंप्यूटर हार्ड डिस्क या माइक्रोकंट्रोलर की मेमोरी में स्टोर होती है। असेंबलर का उपयोग असेंबली प्रोग्राम को मशीन कोड में ट्रांसलेट करने के लिए किया जाता है। असेंबली भाषा में प्रोग्राम लिखने के लिए प्रोग्रामर को सीपीयू या हार्डवेयर का ज्ञान होना चाहिए। निम्न स्तर की भाषाओं का उपयोग क्रॉस डेवलपमेंट में किया जाता है। बाइनरी संख्याओं का प्रतिनिधित्व करने के लिए हेक्साडेसिमल प्रणाली को अधिक कुशल तरीके के रूप में उपयोग किया गया था, जबकि द्विआधारी भाषा का उपयोग करते हुए सीपीयू बहुत तेजी से कार्य करता है।

आज हम कई अलग-अलग प्रोग्रामिंग भाषाओं जैसे—C, JAVA, ORACLE और अन्य का उपयोग कर सकते हैं। इन भाषाओं को उच्च स्तरीय भाषा कहते हैं; प्रोग्राम को उच्च स्तरीय भाषा में लिखने के लिए प्रोग्रामर को हार्डवेयर पर किसी भी ज्ञान की आवश्यकता नहीं होती है जो उच्च स्तरीय एप्लिकेशन के विकास के लिए उपयोग किया जाता है। संकलक उच्च-स्तरीय कार्यक्रम को मशीन स्तर तक अनुवाद करने में महत्वपूर्ण भूमिका निभाता है, क्योंकि मूल विकास में उच्च स्तरीय भाषाओं का उपयोग किया जाता है।

यहाँ कुछ ऐसे टूल्स दिए गए हैं, जो कि माइक्रोकंट्रोलर की प्रोग्रामिंग में उपयोग किए जाते हैं—

- Keil uVison
- Code Editor
- Assembler
- C compiler
- Burner/Programmer

6.3 Keil uVison

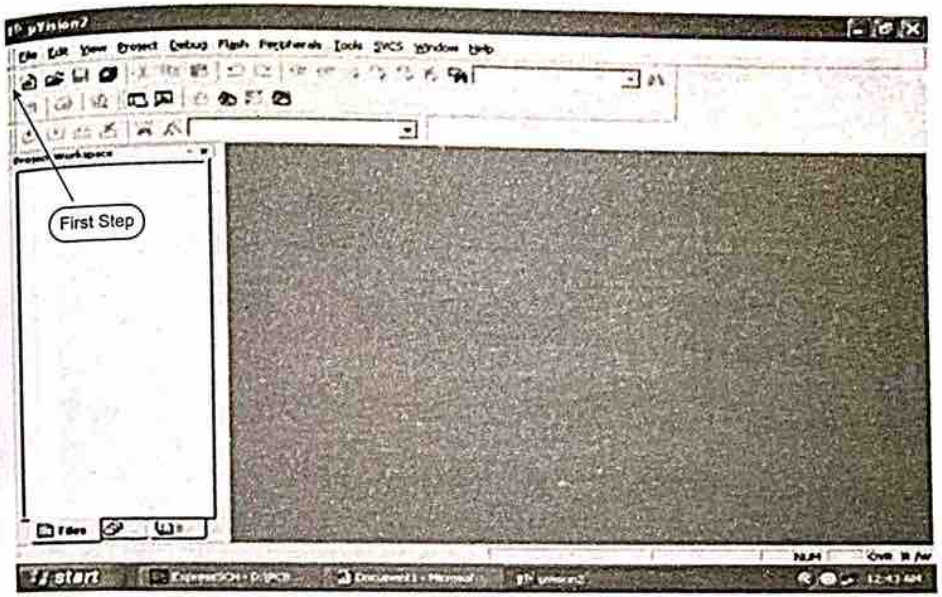
Keil uVison एक नि:शुल्क सॉफ्टवेयर है, जो एम्बेडेड डेवलपर के लिए कई बिंदुओं को हल करता है। यह सॉफ्टवेयर Integrated Development Environment (IDE) है जो प्रोग्राम, एक कंपाइलर लिखने के लिए एक टेक्स्ट एडिटर को एकीकृत करता है और यह सोर्स कोड को हेक्स फाइल में बदल देता है।

चित्र 6.2 : Keil uVsion Software

Keil uVison में कार्य करने के चरण (Steps to Start Working with Keil Uvison)—

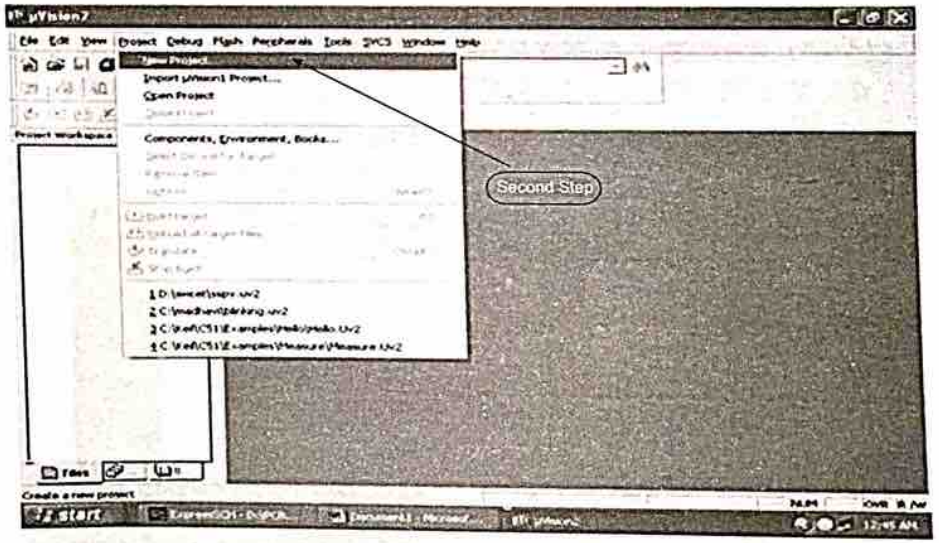
1. डेस्कटॉप पर Keil uVison आइकन पर क्लिक करें।

   इस प्रक्रिया में निम्नलिखित चरण शामिल हैं—



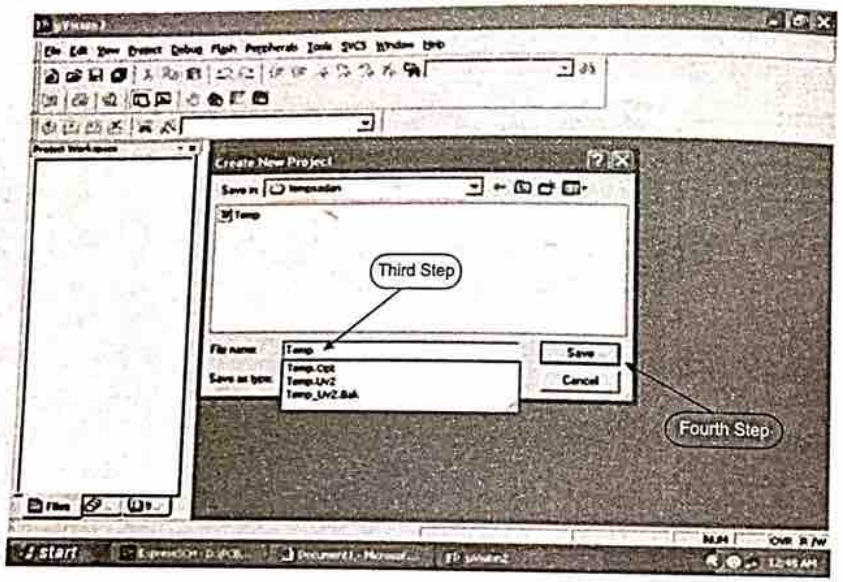
चित्र 6.3

2. Title bar में Project Menu पर क्लिक करें। उसके बाद New Project पर क्लिक करें।



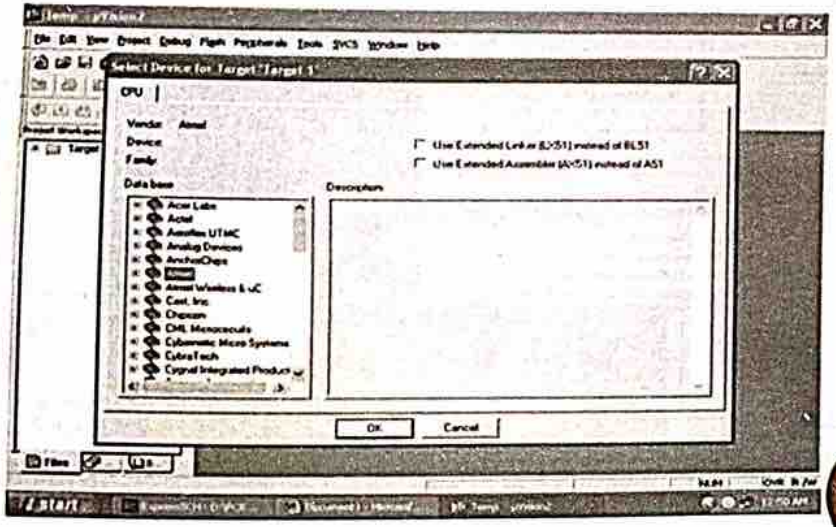
चित्र 6.4

3. प्रोजेक्ट को उपयुक्त नाम देकर अपने फोल्डर में Save करें।



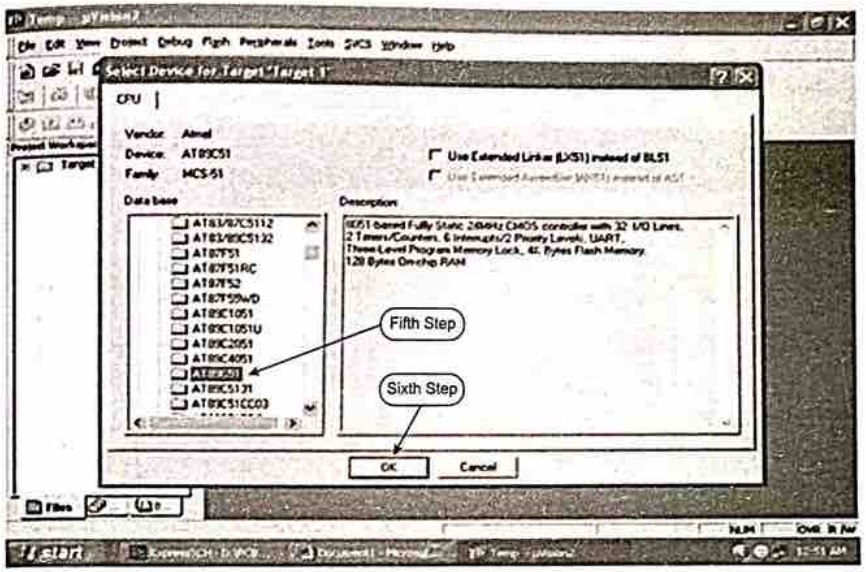
चित्र 6.5

4. Then Click on save button above.
   Select the component for your project. *i.e.* Atmel……
   Click on the + Symbols as for your requirement. Example here selected Atmel.



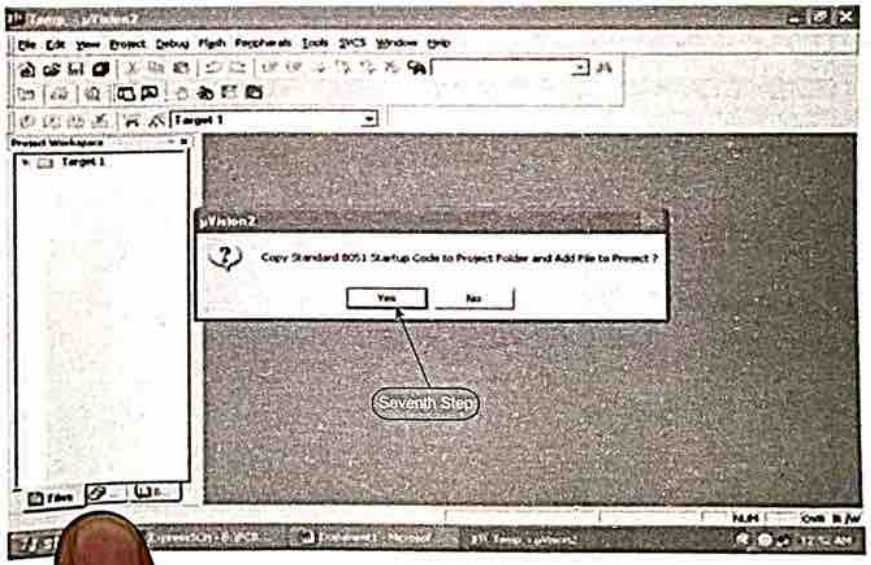
चित्र 6.6

5. Select AT89C51 as shown below



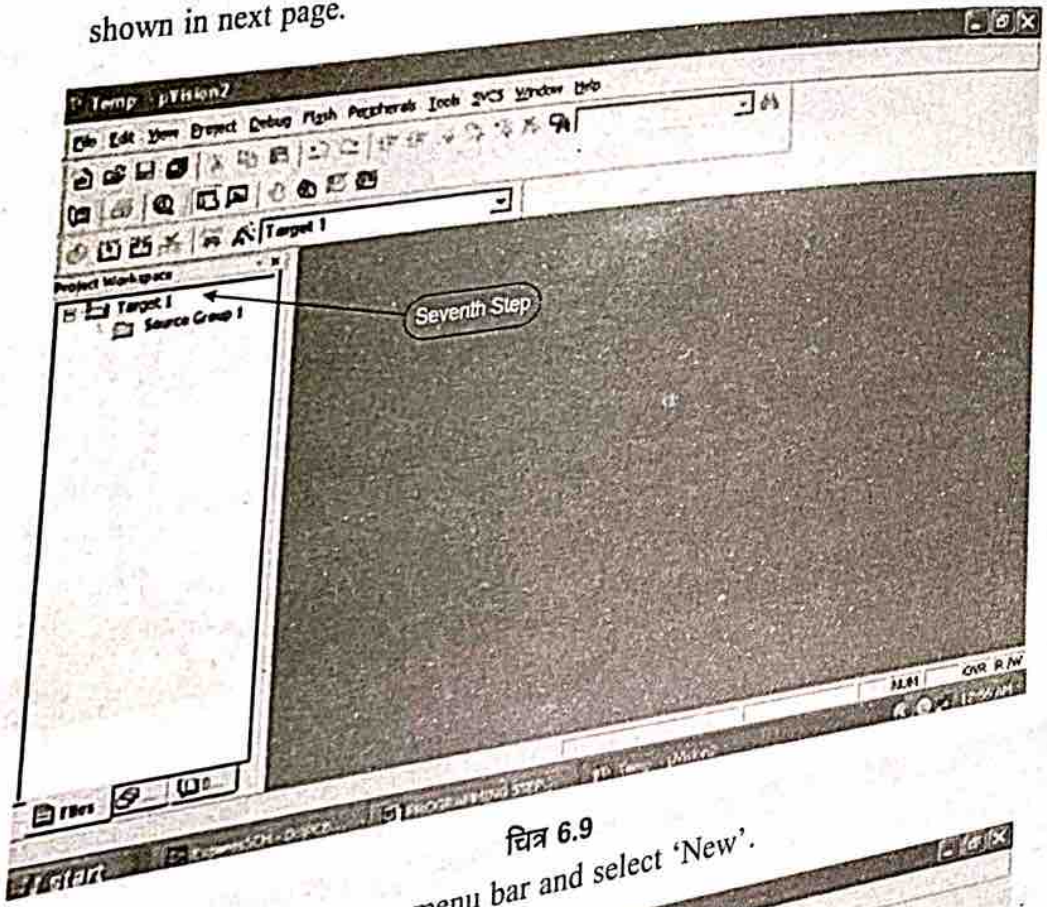
चित्र 6.7

6. Then Click on 'OK'
   The following steps involve in the above process :



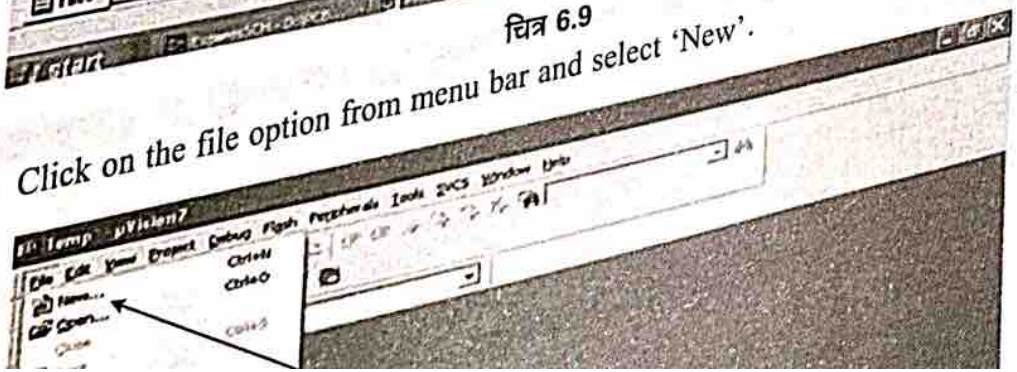
चित्र 6.8

7. Then Click either YES or NO.....mostly 'NO'.
   Now your project is ready to USE.
   Now double click on the Target1, you would get another option 'Source group 1' as shown in next page.

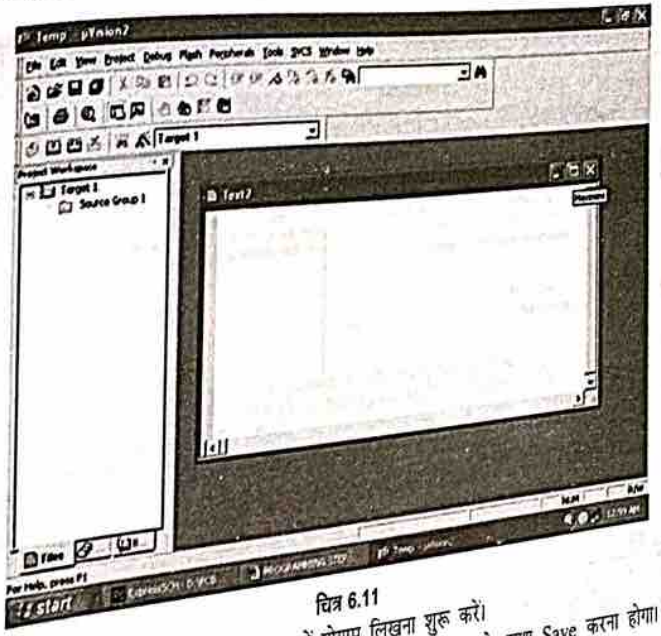


चित्र 6.9

8. Click on the file option from menu bar and select 'New'.

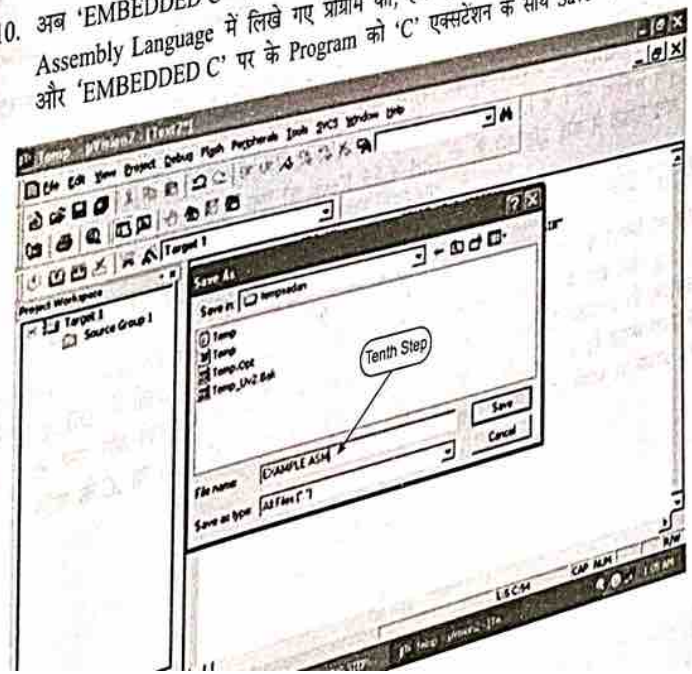


9. The next screen will be as shown in text page.

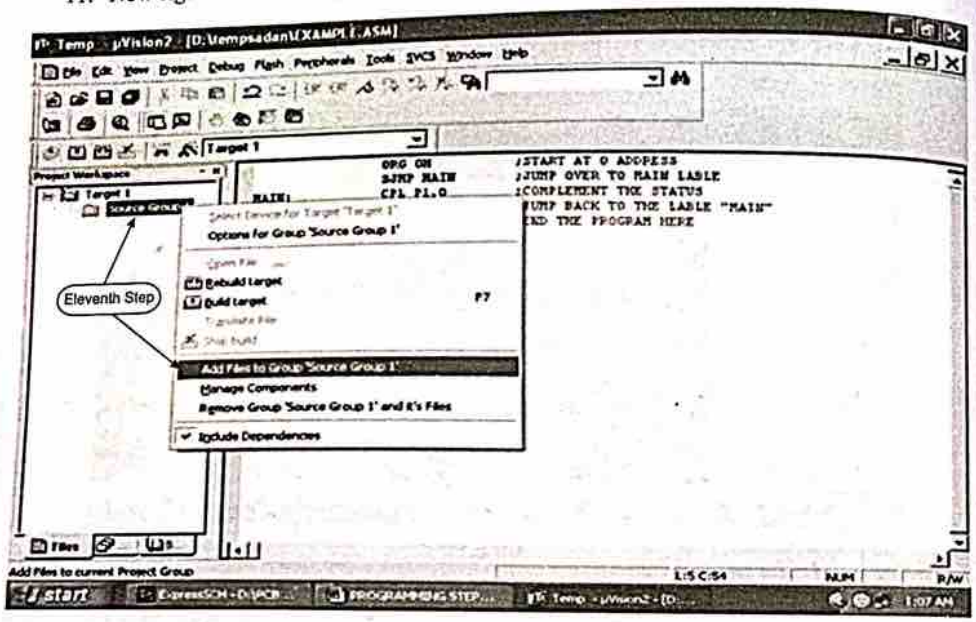


चित्र 6.11

10. अब 'EMBEDDED C' या 'ASM' में प्रोग्राम लिखना शुरू करें।
    Assembly Language में लिखे गए प्रोग्राम को, हमें एक्सटेंशन के साथ Save करना होगा। 'ASM' और 'EMBEDDED C' पर के Program को 'C' एक्सटेंशन के साथ Save करना होगा।



11. Now right click on Source group 1 and click on 'Add files to Group Source'.

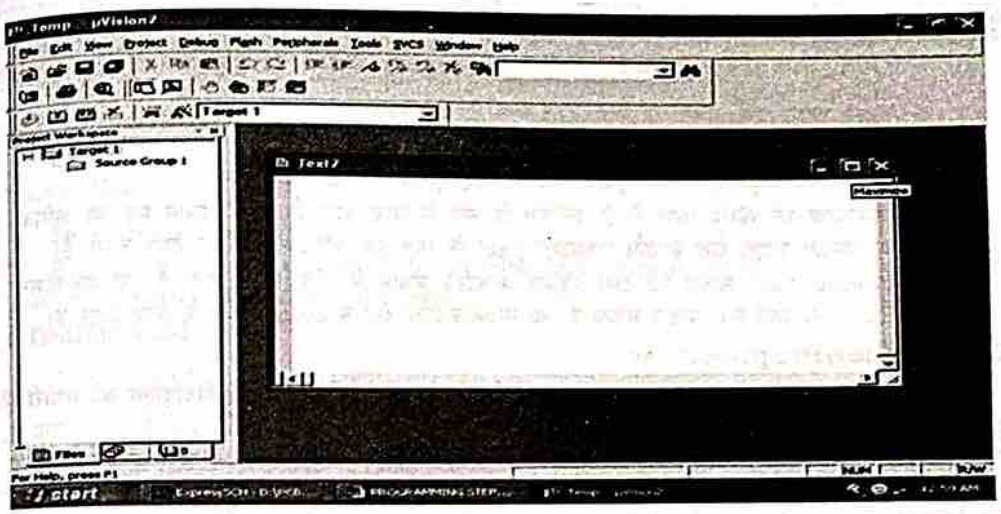


चित्र 6.13

12. अब फाइल को Save करते समय दिए गए अपने फाइल एक्सटेंशन के अनुसार चुनें।
विकल्प 'ADD' पर केवल एक बार क्लिक करें।
अब संकलन करने के लिए फंक्शन कुंजी F7 दबाएँ। ऐसा करने पर कोई त्रुटि दिखाई देगी।
यदि फाइल में कोई त्रुटि नहीं है, तो Ctrl + F5 Keys को एक साथ दबाएं।

6.4 कोड एडिटर या टैक्सट एडिटर (Code Editor or Text Editor)

प्रोग्राम को लिखने के लिए कोड एडिटर का प्रयोग किया जाता है। UVision के Editors में Color syntax, High-lighting जैसी सभी मानक विशेषताएँ शामिल हैं और त्रुटियों को शीघ्रता से पहचानती हैं। DBug करते समय Editor उपलब्ध है। प्राकृतिक Debuging वातावरण आपके प्रोग्राम की त्रुटियों को पहचानने और उन्हें ठीक करने में आपकी सहायता करता है। कोड एडिटर में प्रोग्राम लिखने के बाद उस फाइल को *.asm* या *.C* के फॉर्मेट में Save करें, जिसके आधार पर आपने असेंबलर को चुना है।



चित्र 6.14

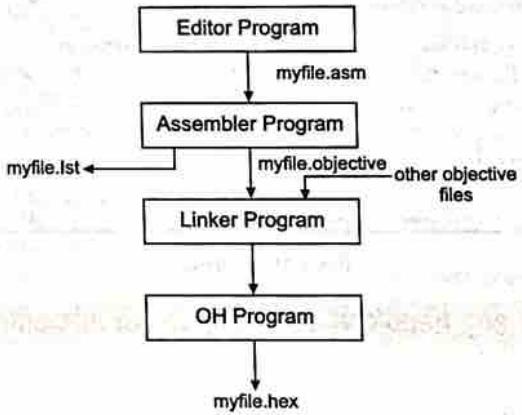
Assembler

असेंबलर का प्रयोग Source Code (निम्न स्तरीय भाषा) को मशीन स्तर (बाइनरी प्रारूप) में बदलने के लिए किया जाता है।

Compiler

संकलक का प्रयोग Source Code (उच्च स्तरीय भाषा) को मशीन स्तरीय (बाइनरी प्रारूप) भाषा में परिवर्तित करने के लिए किया जाता है।

Assembler निर्देशों को मशीन कोड में परिवर्तित करता है—



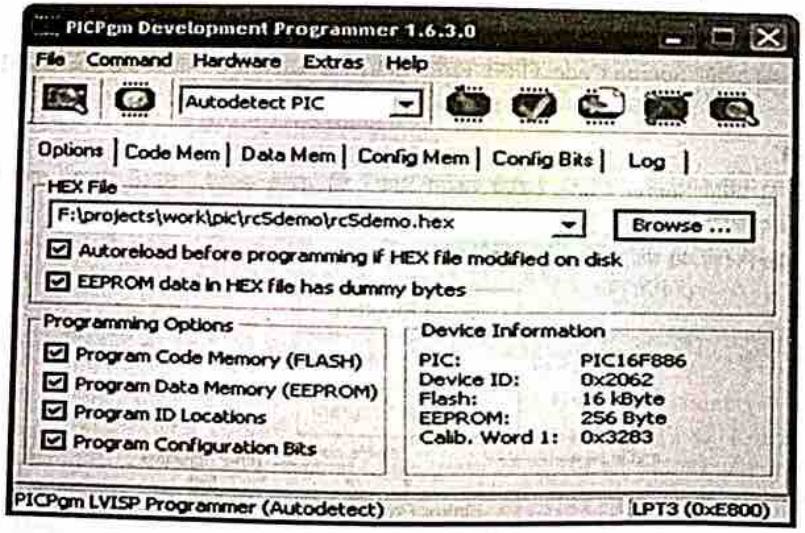
चित्र 6.15 : Compiler

6.5 असेंबली लैंग्वेज से मशीन लेवल में कन्वर्जन (Conversion From Assembly Language to Machine Level)

- पहली फाइल एक एडिटर के साथ बनाई जाती है, जैसे DOS एडिट या अन्य।
- असेम्बलर एक ऑब्जेक्टिव फाइल और फ़ाइल की एक सूची बनाएगा। Object फाइल के लिए एक्सटेंशन '.obj' है, जबकि सूची फाइल के लिए एक्सटेंशन '.lst' है।
- असेम्बलर को तीसरे चरण में है, लिंकिंग के रूप में जाना जाता है। लिंक प्रोग्राम एक या अधिक ऑब्जेक्ट्स फ़ाइल लेता है और एक्सटेंशन '.ab' के साथ एक ऑब्जेक्टिव फाइल तैयार करता है।
- Program '.ab' फाइल को OH (हेक्स कन्वर्टर) प्रोग्राम में फीड किया जाता है, जो एक्सटेंशन '.hex' के साथ एक फाइल बनाता है, जो माइक्रोकंट्रोलर रोम में execute होने के लिए तैयार है।

6.5.1 Burner/Programmer

माइक्रोकंट्रोलर को प्रोग्रामिंग या बर्न करने का अर्थ है **'प्रोग्राम को कंपाइलर से माइक्रोकंट्रोलर की मेमोरी में ट्रांसफर करना'**। माइक्रोकंट्रोलर के लिए Programe मुख्य यप से C या Assembly Language में लिखा जाता है, अंत में कंपाइलर एक हेक्स फ़ाइल बनाता है, जिसमें मशीन भाषा निर्देश होते हैं जैसे शून्य और एक जो माइक्रोकंट्रोलर द्वारा समझा जाता है। यह माइक्रोकंट्रोलर की सामग्री है, जिसे एक बार माइक्रोकंट्रोलर को ट्रांसफर किया जाता है, एक बार प्रोग्राम के अनुसार कार्य करने वाले माइक्रोकंट्रोलर की मेमोरी में इसे ट्रांसफर कर दिया जाता है।



चित्र 6.16 : Burner

6.6 माइक्रोकंट्रोलर और माइक्रोप्रोसेसर के लिए 15 फ्री और ओपन सोर्स सॉफ्टवेयर

1. MIDE-51 Studio

Supported OS : Windows

MIDE-51 is freeware Integrated Development Environment (IDE) for MCS-51 micro-controller.

2. gpsim

Supported OS : Windows, Linux.

gpsim is a full-featured software simulator for Microchip PIC micro-controllers distributed under the GNU General Public License, Version 2 or higher and some of it's libraries under GNU Lesser General Public License, Version 2 or higher.

3. Lab-sticc

Supported OS : Windows.

Power consumption analysis tools for embedded systems. MARTE to AADL model transformation with ATL for tools interoperability.

4. GNUSim 8085

Supported OS : Windows, Linux.

GNUSim8085 is a simulator and assembler for the Intel 8085 Micro-processor.

5. Ktechlab

Supported OS : Linux.

KTechlab is an IDE for micro-controllers and electronics.

6. MC34063 Universal Calculator

A calculation tool for the MC34063.

7. MCU 8051 IDE

Supported OS : Windows, Linux.

MCU 8051 IDE is integrated development enviroment for micro-controllers based on 8051.

8. MSPgcc

Supported OS : Windows, Linux.

mspgcc tool chain provides binutils, gcc, gdb and a lot of other tools for the MSP430 processor.

9. OpenOCD

Supported OS : Windows.

Open On-Chip Debugger provides JTAG/SWD access from GDB (or directly with TCL scripts) to processors with ARM and MIPS based cores.

10. PIC Development Studio

Supported OS : Windows.

PIC Development Studio is a simulator for the PIC16F84 micro-controller. It also provides a plugin framework making it possible to develop custom components. A library of ready-made components is included.

11. PicoForth

Supported OS : Linux.

PicoForth is Forth compiler for PIC12 and PIC16 families. It is written in gForth and requires gpUtils. Produces hex file ready to be programmed into the device.

12. PICsim

Supported OS : Windows

PICsim emulates a micro-controller PIC16F628/16F877A/18F452 and periferics such as USART and timers, the simulator architecture permit easy implementation of external elements in c language. PicsimLab is a realtime emulator of development boards.

13. UrJTAG

Supported OS : Windows.

UrJTAG aims to create an enhanced, modern tool for communicating over JTAG with flash chips, CPUs, and many more. It is a descendant of the popular openwince JTAG tools with a lot of additional features and enhancements.

14. V-USB

Supported OS : Windows, Linux.

V-USB is a software-only implementation of a low-speed USB device for Atmel's AVR micro-controllers, making it possible to build USB hardware with almost any AVR micro-controller, not requiring any additional chip.

15. Yagarto

Supported OS : Windows

YAGARTO is a cross-development environment for the ARM architecture, running on a Windows host. It includes the GNU C/C++ toolchain and the Eclipse IDE.

6.7 एसेम्बली लैंग्वेज प्रोग्रामिंग (Assembly Language Programming)

किसी भी क्षेत्र में 'सम्पर्क बनाने की कला' बहुत-ही महत्वपूर्ण होती है। किन्तु यह तभी सम्भव है, जब सम्पर्क बनाने वाले दोनों ही साथी एक-ही भाषा जानते हों अर्थात् दोनों साथी सम्पर्क के दौरान एक-ही प्रकार के नियमों का पालन कर रहे हों। इन सिद्धान्तों को आरम्भ ध्यान में रखकर हम मानव और माइक्रोकंट्रोलर आपस में सम्पर्क बनाने के लिए जिस भाषा का प्रयोग करते हैं उसे 'एसेम्बली लैंग्वेज (Assembly language)' कहते हैं।

'एसेम्बली लैंग्वेज' नाम 'अग्रेजी या फ्रैंच' की तरह ही है।

जो प्रोग्राम (Program), एसेम्बली लैंग्वेज में लिखे जाते हैं उनको शून्य '0' और वन '1' की भाषा में बदलना अत्यन्त आवश्यक होता है, जिससे कि माइक्रोकंट्रोलर उसे समझ सके।

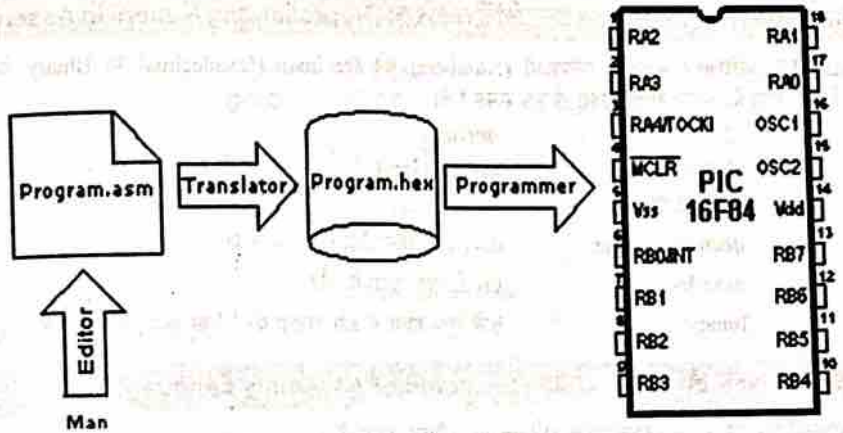
असेम्बली लैंग्वेज (Assembly language) और असेम्बलर (Assembler) दो अलग-अलग शब्द हैं।

असेम्बली लैंग्वेज (Assembly Language)—असेम्बली लैंग्वेज 'नियमों का सेट' (Set of rules) है, जिनका प्रयोग माइक्रोकंट्रोलर के प्रोग्राम (Program) लिखते समय किया जाता है।

असेम्बलर (Assembler)—असेम्बर किसी कंप्यूटर का वह प्रोग्राम है, जो 'असेम्बली लैंग्वेज' को शून्य '0' और एक '1' की भाषा में बदलता है।

6.7.1 मशीन लैंग्वेज (Machine Language)

जो प्रोग्राम '0' और '1' के रूप में प्राप्त होता है, उसे 'मशीन लैंग्वेज (Machine language)' कहते हैं।



चित्र 6.17

वास्तव में 'प्रोग्राम (Program)' कम्प्यूटर की डिस्क (या माइक्रोकंट्रोलर की मेमोरी) में असेम्बलर के नियमों को ध्यान में रखते हुए लिखी गयी 'फाइल' को प्रदर्शित करता है।

असेम्बलर के नियमों के अतिरिक्त भी किसी अन्य माइक्रोकंट्रोलर की भाषा के नियमों के आधार पर भी ये प्रोग्राम लिखे जाते हैं।

मानव केवल असेम्बली लैंग्वेज को ही समझ पाता है, क्योंकि असेम्बली लैंग्वेज में अंग्रेजी वर्णमाला के चिह्न और शब्द होते हैं।

6.7.2 ट्रान्सलेटर (Translater)

ट्रान्सलेटर, असेम्बली भाषा में लिखे गए प्रोग्राम के प्रत्येक निर्देश (Instruction) को '0' और '1' के रूप में परिवर्तित करता है। इन 0 और 1 के रूप को माइक्रोकंट्रोलर आसानी से समझ पाता है।

उदाहरण—माना माइक्रोकंट्रोलर किसी sub-program से वापस मुख्य प्रोग्राम में आने के लिए एक निर्देश (Instruction) 'RETURN' का उपयोग करता है।

जब असेम्बलर इसे ट्रान्सलेट (Translate) करता है, तो हमें 14 bit की '0' और '1' की शृंखला प्राप्त होती है, जिसे माइक्रोकंट्रोलर समझ पाने में सक्षम होता है।

RETURN 00 0000 0000 1000

6.7.3 .EXE File

असेम्बली लैंग्वेज का यह ट्रान्सलेशन/परिवर्तन (Translation) जिस स्थान पर मिलता है, उसे 'एग्जीक्यूशन (Execution) फाइल कहते हैं।

अक्सर हमारा सामना 'HEX' फाइल से होता है। यह उपरोक्त फाइल का Hexadecimal के रूप में प्रदर्शन होता है। **उदाहरण**—'test.hex'।

एक बार यह फाइल Generate हो जाती है, उसके बाद माइक्रोकंट्रोलर द्वारा प्रोग्रामर Programmer की सहायता से इस फाइल को पढ़ा (Read) जाता है।

किसी प्रोग्राम में असेम्बली लैंग्वेज प्रोग्राम को टेक्स्ट प्रोसेसिंग (Text processing) के लिए लिखा जाता है और इस प्रकार यह कम्प्यूटर डिस्क में ASCII file Generate करने में सक्षम होता है।

6.8 असेम्बलर में संख्याओं का प्रदर्शित करना (Representing Nubers in Assembler)

MPLAB असेम्बली भाषा में संख्याओं (Numbers) को Decimal, Hexadecimal या Binary के रूप में प्रदर्शित किया जाता है। आइए संख्या 240 से इसे समझते हैं।

.240	decimal
OxF0	hexadecimal
b'1111 0000'	binary
decimal number	dot (.) से शुरू होती है।
hexadecimal	Ox से शुरू होती है और
binary	b से शुरू होती है और संख्या को ' ' के मध्य रखा जाता है।

6.9 असेम्बली लैंग्वेज के घटक (Elements of Assembly Language)

असेम्बली लैंग्वेज के आधारभूत घटक (Elements) निम्न प्रकार हैं—

1. Labels
2. Instructions
3. Operands
4. Directives
5. Comments

6.9.1 लेबल (Labels)

लेबल, प्रोग्राम में 'किसी एक लाइन' या 'प्रोग्राम के किसी भाग' जहाँ माइक्रो (Micro) jump करके जा सकें' के लिए शाब्दिक पद (Textual designation) होता है। लेबल किसी प्रोग्राम के Set of lines का प्रारम्भ भी हो सकता है।

यह Program branching में भी उपयोग किया जा सकता है; जैसे—Goto ..... ।

इसके अतिरिक्त प्रोग्राम में शर्त के साथ भी Goto instruction execute हो सकता है।

लेबल को Alphabet के किसी Letter अथवा Underline '_____' के साथ शुरू होना अत्यन्त आवश्यक है। लेबल की लम्बाई 32 character तक हो सकती है। लेबल प्रथम कॉलम (First column) से शुरू होना चाहिए।

First column

Correctly written labels

```
Start
_end
P123
Is_it_bigger?
```

Incorrectly written labels

```
 Start    - does not begin in first column
2_end     - begins with a number!
```

चित्र 6.18 : लेबल

6.9.2 निर्देश (Instructions)

माइक्रोकंट्रोलर के अनुसार ही निर्देश (Instruction) लिखे होते हैं। जिस प्रकार से हम Instruction लिखते हैं, उसे 'Instruction syntax' कहते हैं। नीचे सही और गलत तरीके से लिखे गए Instruction दर्शाए गए हैं।

Correctly written instructions

```
movlw   H'01FF'
goto    Start
```

Incorrectly written instructions

```
movlp   H'01FF'
gotto   Start
```

चित्र 6.18 : Instruction

6.9.3 ऑपरैंड (Operands)

Operand, जो निर्देश (Instruction) Execute होती है उसके लिए Instruction element होता है। सामान्यतया ये Registers या Variables या Constants होते हैं।

Basic information on the program

```
; Program for initialization of port B and setting pins to status of logic one
; Version 1.0  Date: 10.10.1999.  MCU:PIC16F84  Written by: John Smith

; Declaration and configuration of a processor
        PROCESSOR 16F84
        #include  "p16f84.inc"  ; Processor title

        __CONFIG  _CP_OFF & _WDT_OFF & _PWRTE_ON & _XT_OSC

                                ; Start of program
        org     0x00            ; Reset vector
        goto    Main            ; Go to the beginning of Main

                                ; Interrupt vector
        org     0x04            ; Interrupt vector
        goto    Main            ; Interrupt routine doesn't exist
        #include "bank.inc"
                                ; Beginning of the main program
Main
        BANK1                   ; Select memory bank 1
        movlw   0x00
        movwf   TRISB           ; Port B pins are output
        BANK0                   ; Select memory bank 0

        movlw   0xFF
        movwf   PORTB           ; Set all ones to port B

Loop    goto    Loop            ; Program remains in the loop

        end                     ; Necessary marking the end of a program
```

Directive
Inclusion of a macro
Comment
Label
Instruction
Operand

चित्र 6.19 : Operands

6.9.4 डाइरेक्टिव्स (Directives)

Directives भाग एक प्रकार के Instruction ही होते हैं, लेकिन ये माइक्रोकंट्रोलर पर निर्भर नहीं करते हैं। ये Instruction असेम्बलर के लिए होते हैं।

Directive को Variables या Registers के उपयोग द्वारा अर्थपूर्ण बनाया जाता है।

उदाहरण—RAM मेमोरी में, LEVEL किसी Variable के लिए, Address ODR प्रदर्शित करता है। इस प्रकार उस Address के Variable को LEVEL पद द्वारा प्रदर्शित करते हैं। इस प्रोग्रामर को ODR याद रखने की बजाय LEVEL याद रखकर लिखना आसान होता है।

कुछ बहुतायत में प्रयुक्त होने वाले Directive इस प्रकार हैं—

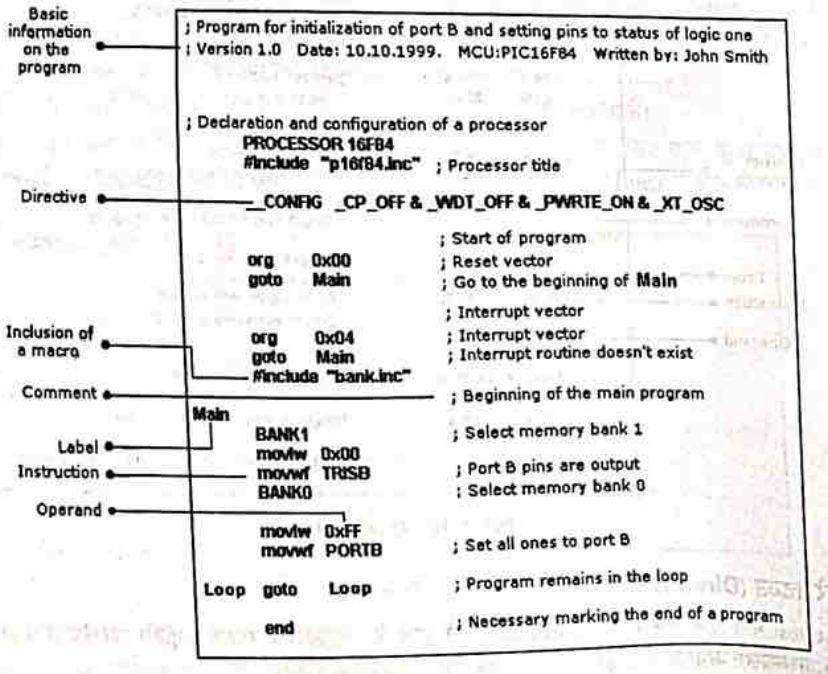
```
PROCESSOR        16F84
#include         "p16f84.inc"
__CONFIG__CP__OFF&__WDT__OFF&___PWRTE___ON&___XT___OSC
```

कमेंट्स (Comments)

किसी प्रोग्राम को अच्छी प्रकार से समझने योग्य बनाने के लिए प्रोग्रामर द्वारा Comments लिखे जाते हैं। इसे किसी Instruction के बाद "; " सेमीकॉलन लगाकर लिखा जाता है।

6.10 सैम्पल प्रोग्राम लिखना (Writing a Sample Program)

नीचे Basic rules का पालन करते हुए एक Simple program लिखा गया है। Basic rules के पालन के अतिरिक्त भी कुछ नियमों का पालन करना चाहिए, जैसे—प्रारम्भ में "प्रोग्राम का नाम" लिखना चाहिए, प्रोग्राम क्या कार्य कर रहा है, Version, date of writing the program, किस माइक्राकंट्रोलर के लिए यह प्रोग्राम लिखा गया है और प्रोग्राम लिखने वाले (प्रोग्रामर) का नाम लिखना उचित होता है। क्योंकि ये सूचनाएँ असेम्बली ट्रान्सलेटर के लिए महत्वपूर्ण नहीं होती हैं इसलिए इन्हें 'comments' के रूप में लिखना अत्यन्त आवश्यक होता है। ध्यान दें कि Comments हमेशा सेमीकॉलन (;) के साथ शुरू होता है।



चित्र 6.20 : Sample Program

Comments को नई लाइन से भी लिखा जा सकता है अथवा किसी के बाद में भी लिखा जा सकता है।

जब आरम्भ में Comments लिखे जा चुके हों, तो उसके बाद Directive अवश्य लिखने चाहिए। आगे उदाहरण में यह दर्शाया गया है।

माइक्रोकंट्रोलर के पैरामीटर को परिभाषित करना अत्यन्त आवश्यक होता है, जिससे लिखा गया प्रोग्राम सही प्रकार से कार्य कर सके। माइक्रोकंट्रोलर पैरामीटर जैसे कि—

- type of oscillator,
- whether watch dog timer is turned ON किया watch dog timer ऑन किया गया है।
- whether internal reset circuit is enabled

उपरोक्त सभी निम्न प्रकार के Directive को इस प्रकार लिखकर परिभाषित किया जाता है—

__CONFIG__CP__OFF&__WDT__OFF&___PWRTE___ON&XT___OSC

जब सभी आवश्यक घटकों को परिभाषित कर दिया जाता है, उसके बाद प्रोग्राम लिखना शुरू करना चाहिए।

सबसे पहले यह जानना आवश्यक होता है, कि माइक्रोकंट्रोलर किस Address से शुरू होता है और उसके बाद Power Supply Start-up शुरू होना चाहिए। यह Address है—(ORG 0x00)

यदि Interrupt आता है तो प्रोग्राम लिख रहे हैं इसलिए माइक्रोकंट्रोलर को प्रोग्राम "goto Main" Instruction से प्रारम्भ करना उचित होगा।

'Main' में लिखी गयी Instruction, memory bank 1 (Bank 1) को चुनती हैं, जिससे TRISB register तक पहुँचा जा सके। इसलिए Part B को output port घोषित करना चाहिए।

(mov *lw* 0x00, mov wf TRISB).

अगला स्टेप है—मेमोरी हैक 0 को चुनना और Port B पर Logic '1' status रखना।

(mov *lw* 0xFF, mov wf Port B).

इस प्रकार मुख्य प्रोग्राम समाप्त होता है।

हमें एक Loop और बनाना चाहिए, जिसमें micro रखे जाएंगे। इस प्रकार यह भटकता (Wander) नहीं है यदि कोई त्रुटि होती है। इस उद्देश्य के एक Infinite loop बनाया जाता है, जिसमें Micro रखते हैं और Power चालू रहती है। प्रोग्राम के अन्त में 'end' अवश्य लिखा जाता है, जिससे Assembly translator को यह पता चलता है, कि अब आगे और कोई Instruction नहीं लिखी है।

6.11 Control Directives

6.11.1 #DEFINE : Exchanges one part of text for anoter

Syntax :

#define<text> [<another text>]

Description :

Each time <text> appears in the program, it will be exchanged for <another text>.

Example :

```
#define turned_on 1
#define turned_off 0
```

Similar directives : # UNDEFINE, IFDEF, IFNDEF.

6.11.2 INCLUDE : Include an additional file in a program

Syntax :

#include <file_name>

#include "file_name"

Description : An application of this directive has the effect as though the entire file was copied to a place where the "include" directive was found. If the file name is in the square brackets, we are dealing with a system file, and if it is inside quotation marks, we are dealing with a user file. The directive "include" contributes to a better layout of the main program.

Example :

include <regs . h>

#include "subprog.asm"

6.11.3 CONSTANT : Gives a constant numeric value to the textual designation

Syntax :

Constant <name> = <value>

Description : Each time that <name> appears in program, it will be replaced with <value>.

Example :

Constant MAXIMUM=100

Constant Length=30

Similar directives : SET, VARIABLE

6.11.4 VARIABLE : Gives a variable numeric value to textual designation

Syntax :

Variable<name> = <value>

Description : By using this directive, textual designation changes with particular value.

It differs from CONSTANT directive in that after applying the directive the value textual designation can be changed.

Example :

variable level=20

variable time=13

Similar directives : SET, CONSTANT

6.11.5 SET : Defining assembler variable

Syntax :

<name - under score variable>set<value>

Description : To the variable <name_variable> is added expression <value>. SET directive similar to EQU, but with SET directive name of the variable can be redefined following a definition.

Example :

level set 0

length set 12

level set 45

Similar directives : EQU, VARIABLE

6.11.6 EQU : Defining assembler constant

Syntax :

<name_constant> equ <value>

Description : To the name of a constant <name-constant> is added value <value>

Example :

five equ 5

six equ 6

seven equ 7

Similar instructions : SET

6.11.7 ORG : Defines an address from which the program stored in micro-controller memory

Syntax :

<iabel>org<value>

Description : This is the most frequently used directive. With the help of this directive we, define where some part of a program will be start in the program memory.

Example :

```
Start org 0x00
      movlw 0xFF
      movwf PORTB
```

The first two instructions following the first 'org' directive are stored from address 00, and the other two from address 10.

6.11.8 END : End of program

Syntax :

end

Description : At the end of each program it is necessary to place 'end' directive, so that assembly translator would know that there are no more instructions.

Example :

```
.
.
movlw 0xFF
movwf PORTB
end
```

6.11.4 IF : Conditional program branching

Syntax :

if<conditional_term>

Description : If condition in <conditional_term> was met, part of the program which, IF directive would be executed. And if it wasn't, then the part following ELSE or ENDIF directive would be executed.

Example :

if level=100

```
goto FILL
else
goto DISCHARGE
endif
```

Similar directives : #ELSE, ENDIF.

6.11.10 ELSE : The alternative to 'IF' program conditional terms

Syntax :

Else

Description : Used with IF directive as an alternative if conditional term is incorrect.

Example :

```
If time< 50
goto SPEED UP
else goto SLOW DOWN
endif
```

Similar instructions : ENDIF, IF

6.11.11 ENDIF : End of conditional program section

Syntax :

endif

Description : Directive is written at the end of a conditional block to inform the translator that it is the end of the conditional block.

Example :

```
If level=100
goto LOADS
else
goto UNLOADS
endif
```

Similar directives : ELSE, IF

6.11.12 WHILE : Execution of program section as long as condition is met

Syntax :

while<condition>
endw

Description : Program lines between WHILE and ENDW would be executed as long as condition was met. If a condition stopped being valid, program would continue executing instructions following ENDW line. Number of instructions between WHILE and ENDW can be 100 at the most, and number of executions 256.

Example :

```
While i<10
i=i+1
endw
```


6.11.13 ENDW : End of conditional part of the program

Syntax :

endw

Description : Instruction is written at the end of the conditional WHILE block, so that assembly translator would know that it is the end of the conditional block.

Example :

```
while i<10
i=i+1
endw
```

Similar directives : WHILE.

6.11.14 IFDEF : Execution of a part of the program if symbol was defined

Syntax :

ifdef<designation>

Description : If designation <designation> was previously defined (most commonly by #DEFINE instruction), instructions which follow would be executed until ELSE or ENDIF directives would be reached.

Example :

```
#define test
.
ifdef test ;how the test was defined
.....; instructions from these lines would execute
endif
```

Similar directives : #DEFINE, ELSE, ENDIF, IFNDEF, #UNDEFINE

6.11.15 IFNDEF : Execution of a part of the program if symbol was defined

Syntax :

ifndef<designation>

Description : If designation <designation> was not previously defined, or if its definition was erased with directive #UNDEFINE, instructions which follow would be execute until ELSE or ENDIF directive would be reached.

Example :

```
#define test
.....
#undefine test
.....
ifndef test ;how the test was undefined
.....; instructions from these lines would execute
endif
```

Similar directives : #DEFINE, ELSE, ENDIF,! IFDEF, #UNDEFINE.

6.11.16 CBLOCK : Defining a block for the named Constants

Syntax :

Cblock [<term>]
<label>[:<increment>], <label>[:<increment>].....

endc

Description : Directive is used to give values to named constants. Each following term receives a value greater by one than its precursor. If <increment> parameter is also given, then value given in <increment> parameter is added to the following constant.

Value of <term> parameter is the starting value. If it is not given, it is considered to be zero.

Example :

```
Cblock 0x02
First, second third ;first=0x02, second=0x03, third=0x04
endc
cblock 0x02
first : 4, second : 2, third ; first=0x06, second=0x08, third=0x09
endc
```

Similar directives : ENDC

6.11.17 ENDC : End of constant block definition

Syntax :

endc

Description : Directive was used at the end of a definition of a block of constants, so assembly translator could know that there are no more constants.

Similar directives : CBLOCK

1.11.18 DB : Defining one byte data

Syntax :

[<label>]db <term> [, <term>,.....,<term>]

Description : Directive reserves a byte in program memory. When there are more terms which need to be assigned a byte each, they will be assigned one after another.

Example :

db 't', 0xOf, 'e', 's', 0×12

Similar instructions : DE, DT

1.11.19 DE : Defining the EEPROM memory byte

Syntax :

[<term>] de <term> [, <term>,.....,<term>] ;

Description : Directive is used for defining EEPROM memory byte. Even though it was first intended only for EEPROM memory, it could be used for any other location in any memory.

Example :

org H'2100'

de "Version 1.0", 0

Similar instructions : DB, DT

1.11.20 DT : Defining the data table

Syntax :

[<label>] dt <term> [, <term>,....., <term>]

Description : Directive generates RETLW series of instructions, one instruction per each terms.

Example :

dt "Message", 0

dt first, second, third

Similar directives : DB, DE

6.11.21 CONFIG Setting the configuration bits

Syntax :

__config<terrm> or__ _config<address>,<term>

Description : Oscillator, watchdog timer application and internal reset circuit are defined. Before using this directive, the processor must be defined using PROCESSOR directive.

Example :

_CONFIG _CP_OFF&_WDT_OFF&_PWRTE_ON&_XT_OSC

Similar directives : _IDLOCS, PROCESSOR.

6.11.22 PROCESSOR : Defining micro-controller model

Syntax :

Processor <micro-controller_type>

Description : Instruction sets the type of micro-controller where programming is done.

Example :

processor 16F84

6.11.13 Files created as a result of program translation

As a result of the process of translating a program written in assembler language we get files like :

- Executing file (Program_Name.HEX)
- Program errors file (Program_Name.ERR)
- List file (Program_Mame.LST)

The first file contains translated program, which was read in micro-controller by programming. Its contents cannot give any information to programmer, so it will not be considered any further.

The second file contains possible errors that were made in the process of writing and which were noticed by assembly translator during translation; process Errors can be discovered in a 'list' file as well. This file is more suitable though when program is big and viewing the 'list' file takes longer.

The third file is the most useful to programmer. Much information is contained in it, like information about positioning instructions and variables in memory, or eror signalization.

Example of 'list' file for the program in this chapter follows. At the top of each page is stated information about the file name, date when it was translated and page number. First column contains an address in program memory where an instruction from that row is placed. Second column contains a value of any variable defined by one of the directives : SET, EQU, VARIABLE, CONSTANT or CBLOCK. Third column is reserved for the form of a translated instruction which PIC is executing. The fourth ; column contains assembler instructions and programmer's comments. Possible errors will appear between rows following a line in which the error occurred.

Makro: Proba.lst

```
MPASM 02.40Released          PROBA.ASM   4-26-2000  7:18:17          PAGE  1

LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

                 00001  ;Program for initialization of port B and setting its pins
                 00002  ;to the state of logic one
                 00003  ;Version: 1.0  Date: 10.05.2000.     MCU: PIC16F84  Written
                 00004  ;by: Petar Petrovic
                 00005
                 00006  ;Declaration and configuration of the processor
                 00007  PROCESSOR 16F84
                 00008  #include "p16f84.inc"  ;Processor title
                 00001  LIST
                 00002  ;P16F84.INC  Standard Header File, Version 2.00  Microchip
                        ;Technology, Inc.
                 00136  LIST
                 00009
2007 3FF1        00010  __CONFIG   _CP_OFF & _WDT_OFF & _PWRTE_ON & _XT_OSC
                 00011
000C             00012  CONSTANT BASE = 0x0c
                 00013
                 00014  ;Start of a program
0000             00015  org  0x00       ;Reset vector
0000 2805        00016  goto  Main      ;Go to the beginning of the main program
                 00017
                 00018  ;Interrupt vector
0004             00019  org      0x04   ;Interrupt vector
0004   2805      00020  goto     Main   ;Interrupt routine does not exist
                 00021
                 00022 ;Beginning of the main program
                 00023  #include "Bank.inc"     ; File with macros
                 00001 ;*************************************************************
                 00002 ;          Makros BANK0 and BANK1
                 00003 ;*************************************************************
                 00004
0000   0010      00005  W_Temp          set     BASE+4
0000   0011      00006  Stat_Temp       set     BASE+5
0000   0012      00007  Option_Temp     set     BASE+6
                 00008
                 00009
                 00010  BANK0   macro
                 00011  bcf     STATUS,RP0      ; Select memory bank 0
                 00012  endm
                 00013
                 00014  BANK1   macro
                 00015  bsf     STATUS,RP0      ; Select memory bank 1
                 00016  endm
                 00017
0005             00024  Main
                 00025  BANK1                   ; Select memory bank 1
0005   1683      M      bsf     STATUS,RP0      ; Select memory bank 1
0006   3000      00026  movlw   0x00
Message[302]: Register in operand not in bank 0. Ensure that bank bits are
correct.
0007   0086      00027  movwf   TRISB           ;Port B pins are output
                 00028
                 00029  BANK0                   ;Select memory bank 0
0008   1283      M      bcf     STATUS,RP0      ;Select memory bank 0
0009   30FF      00030  movlw   0xFF
000A   0086      00031  movwf   PORTB           ;Set all ones to port B
                 00032
000B   280B      00033  Loop    goto    Loop    ; Program stays in the loop
                 00034
                 00035  END                     ;Necessary marking the end of a program

MEMORY USAGE MAP ('X' = Used,  '-' = Unused)

0000 : X---XXXXXXXX---- ---------------- ---------------- ----------------
2000 : -------X-------- ---------------- ---------------- ----------------

All other memory blocks unused.

Program Memory Words Used:    9
Program Memory Words Free:    1015

Errors:        0
Warnings:      0 reported,      0 suppressed
Messages:      1 reported,      0 suppressed
```

At the end of the "list" file there is a table of symbols used in a program. Useful element of 'list' file is a graph of memory utilization. At the very end, there is an error statistic as well as the amount of remaining program.

Table 1 Assembler Directives

Name of Assember Directive	what it does?	Alias for	
END	end program		
DB	define bytes	FCB	
DW	define words	FDB	
DS	define sotrage	RMB	
EQU	equate		
FCB	form constant byte		
FCC	from constant characters		
FDB	form double bytes		
ORG	set origin		
RMB	reserve memory bytes		
#INCLUDE	include source file		
$INCLUDE	include source file	#INLUDE	

The <OPERAND> contains a value, an expression, an address, or a label that the opcodes or the directives need. The operand could be up to 4 bytes long, separated by commas. Some opcodes or directives do not require operands (inherent mode).

The constants used in hex, decimal, binary, or octal numbers. Table 2 gives the assembler symbols used to this purpose.

Table 2 : Assembler synaibois for coastants

Symbol	Meaning	Example
$<number>	hex number	$A1
<number>	decimal number	20
%<number>	binary bumber	%11001010
@<number>	octal number	@73
'<string>', '<string>'	ASCII string	'A' or 'A (the latter does not work with #INCLUDE)

The expressions used any of the operators listed in Table 3

Table 3 : Assembler symbols for constants

Symbol	Meaning	Example
-	unary minus	-4
&	binary AND	%11111111&%10000000
!	binary OR	%11111111!%10000000
*	multiplication	3*$2A
/	division	$7E/3

+	addition	1+2
-	subtraction	3-1
()	parentheses used for grouping	3*(1+2)

Important convenstons used are given in Table 4 :

Table 4 : Assembler symbols for coastants

Symbol	Meaning	Example
#	immediate mode (IMM)	#$A3
;	start of comment line and of comment inside a program statement	LDAA #$FF; Load accA
*	alternate sign for start of comment line only	*This is a comment
,X	index X mode (IND,X)	LDAA TFLG1,X
,Y	index X mode (IND,Y)	LDAA TFLG2,X

The <LABEL> is a very powerful concept that can greatly simplify the programmer's task. The <LABEL> consists of a string of alphanumeric characters that make up a name somehow meaningful to , the programmer. The placement of the <LABEL> can be in one of the following positions :

1. In the first column and terminates with a tab or blank character.
2. In any column and terminates with a colon (:)

There are three different usages of the <LABEL> :

(i) To assign the name inserted in the <LABEL> to a location in a program. The <LABEL> will be assigned the address of that location.

(ii) To assign the value of an expression or constant to the name inserted in the <LABEL> using the EQU (equate) or SET directives.

(iii) To define the name of a subroutine (macro). Essentially, this is the same as 1), since an address (the subroutine starting address) is assigned to the label.

When labels are assigned to certain addresses, one can tell the program to go to that address by referring to the label (case 1 above). Alternatively, one can use the contents of a certain address by referring to its label, just like when using variables (case 2 above).

A comment is prefixed by semicolon (;).When the assembler detects an semi-colon, it knows that the rest of the line is a comment and does not expect any executable instructions from it. A comment can be a separate line (comment line) or can be inserted in a program statement. A comment line can be also prefixed by an asterisk (*). The comments, either in the comment field or as a separate comment line are of great benefit to the programmer in debugging, maintaining, or upgrading a program. A comment should be brief and specific and not just reiterate its operation. A comment that does not convey any new information needs not be inserted. When writing a comment, use lower case characters.

A program written in Assembly language is called source file. Its extension is *.ASM*. When the source file is assembled, two files are generated :

1. Object file that can be run in the micro-controller. The Motorola object file is in ASCII-HEX format. Its generic name is 'S19 file'. Its extension is *.S19*

2. List file, extension *.LST*, that contains the original code in Assembly language and the corresponding hex codes resulting from the Assembly process. The list file is used by the programmer to verify and debug his/her coding of the program.

The *.ASM* files can be opened, viewed, edited and saved in the THRSIM11 application. Alternatively, all three file types (*.ASM, .1ST, .S19*) can be also processed in a text editor, *e.g.*, the Notepad application.

Examples of *.ASM* and *.1ST* files follow.

Addressing Modes : Inherent Mode is implied and requires no programming action.

Immediate Mode means that the number contained in the operand will be immediately used.

Direct and Extended Modes use the number contained in the operand to signify an address where the required information should be retrieved from or deposited to. The Extended mode is automatically used for addresses greater than FF.

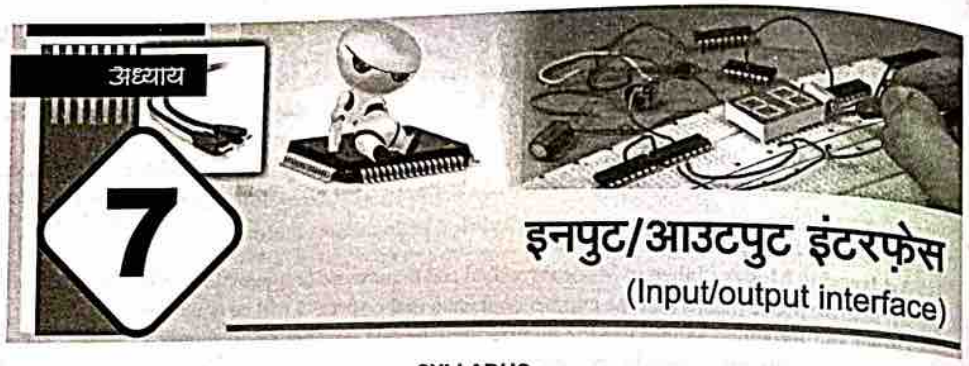
Index Mode is used by adding the operand to the value already existing in the Index X or Y, as selected. In this case, the operand acts as an offset.

Relative Mode uses the operand as an offset relative to the present Program Counter value.

प्रश्नावली

1. माइक्रोकंट्रोलर में प्रोग्राम लिखने के मूलभूत नियमों का वर्णन कीजिए।
2. माइक्रोकंट्रोलर में किस प्रकार का सॉफ्टवेयर प्रयोग किया जाता है?
3. माइक्रोकंट्रोलर में C या ASM code किस प्रकार ट्रांसफर (Transfer) किए जाते हैं?
4. एसेम्बली लैंग्वेज प्रोग्राम क्या होते हैं?
5. निम्न पदों की व्याख्या कीजिए—
   1. मशीन लैंग्वेज
   2. एसेम्बली लैंग्वेज
   3. ट्रांसलेटर
   4. Mnemonics
   5. Directives
6. असेम्बली लैंग्वेज में प्रोग्राम लिखते समय किन-किन बातों का ध्यान रखना चाहिए (Basic rules के अतिरिक्त)?
7. 8051 माइक्रोकंट्रोलर में Analog data को Digital data में बदलने के लिए ADC प्रोग्राम लिखिए।

❑



इनपुट/आउटपुट इंटरफ़ेस
(Input/output interface)

SYLLABUS

Sensors, 7-segment display, LCD, LED and relay.

7.1 सेंसर (Sensor)

सेंसर एक ऐसा उपकरण है, जो अपने वातावरण से भौतिक इनपुट को मापकर ऐसे डाटा में परिवर्तित करता है, जिसकी व्याख्या मानव या मशीन द्वारा की जा सकती है। अधिकांश सेंसर इलैक्ट्रॉनिक होते हैं (डाटा को इलैक्ट्रॉनिक डाटा में परिवर्तित किया जाता है), लेकिन कुछ अधिक सरल होते हैं, जैसे—ग्लास थर्मामीटर, जो दृश्य डाटा प्रस्तुत करता है। सेंसर एक विद्युत उपकरण होता है। सेंसर की मदद से हम किसी भी वस्तु की पूरी जानकारी पता कर सकते है।

जैसे—उस वस्तु की ऊँचाई, वजन, दूरी, तापमान और भी कई जानकारी हमे मिलती है।

7.2 सेंसर के प्रकार (Types of Sensor)

1. टेम्प्रेचर सेंसर (Temperature sensor)
2. इंफ्रारेड सेंसर (Infrared sensor)
3. प्रोक्सिमिटी सेंसर (Proximity sensor)
4. प्रेशर सेंसर (Pressure sensor)
5. लेवल सेंसर (Level sensor)
6. स्मोक और गैस सेंसर (Smoke and Gas sensor)
7. अल्ट्रासोनिक सेंसर (Ultrasonic sensor)

7.2.1 टेम्प्रेचर सेंसर (Temperature Sensor)

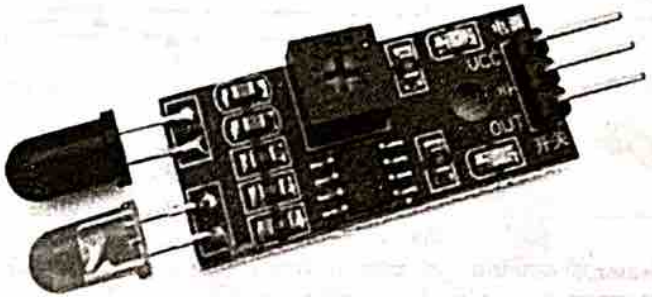
टेम्प्रेचर सेंसर का उपयोग किसी भी वस्तु के तापमान को मापने के लिए किया जाता है; जैसे—यदि हमें किसी गिलास में रखे पानी का तापमान पता करना है, तो इसके लिए हम टेम्प्रेचर सेन्सर का उपयोग करते हैं।

कार्य करने का तरीका—इस सेंसर में थरमिस्टर का भी उपयोग किया जाता है। टेम्प्रेचर सेंसर को एक मेटल से जोड़ दिया जाता है। तापमान कम या अधिक होने पर मेटल के रेजिस्टेंस में परिवर्तन होता है, और उस रेजिस्टेंस के परिवर्तन की मदद से ही यह तापमान का पता लगता है।

चित्र 7.1 : टेम्प्रेचर सेंसर

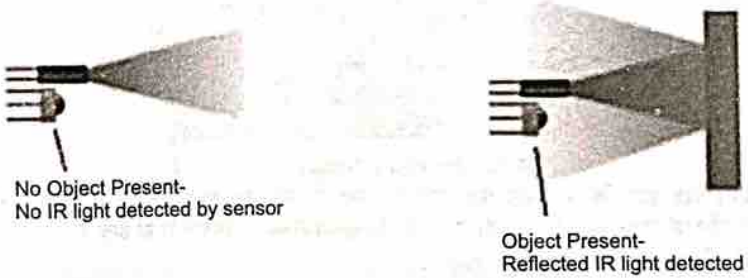
7.2.2 इंफ्रारेड सेंसर (Infrared Sensor)

इस सेंसर को IR sensor भी कहते हैं। यह एक इलैक्ट्रॉनिक उपकरण होता है। इन सेन्सर का उपयोग वायरलेस सिस्टम और रिमोट कन्ट्रोल में अधिक किया जाता है। IR sensor में दो भाग होते हैं। एक IR Transmitter व दूसरा IR Receiver होता है।



चित्र 7.2 : IR sensor मोड्यूल

IR sensor के कार्य करने का तरीका—यदि आपने कभी IR sensor को देखा है, तो आपको इसमे 2 LED दिखेगी। इन दोनो का अलग-अलग कार्य होता है।

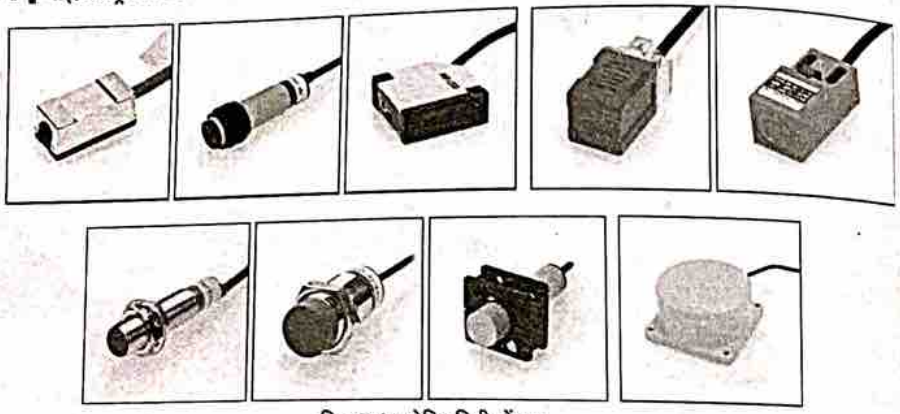


चित्र 7.3

इनमे से एक LED में ट्रांसमीटर होता है, जो की वायु में रेडिएशन को भेजने का कार्य करता है। इसके बाद में दूसरा कार्य एक रिसीवर डिवाइस का होता है। ट्रांसमीटर से भेजी गयी रेडिएशन रिसीवर के पास आ जाती है। यदि कोई वस्तु इन रेडिएशन के बीच में आती है, तो उस समय रेडिएशन रिसीवर तक नही पहुँच पाती है और इस तरह IR sensor कार्य करता है।

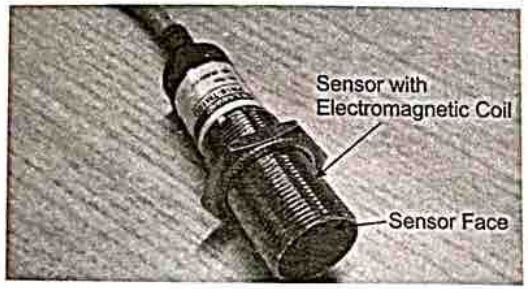
7.2.3 प्रोक्सिमिटी सेंसर (Proximity sensor)

कम्पनी में या फिर अन्य किसी स्थान पर आपने यह सेंसर सबसे ज्यादा देखे होंगे। प्रोक्सिमिटी का अर्थ 'पास होना' होता है।



चित्र 7.4 : प्रोक्सिमिटी सेंसर

Proximity Sensor की कार्यविधि—यदि प्रोक्सिमिटी सेंसर के निकट कोई वस्तु आती है तो यह सेन्सर उसको सेन्स कर लेता है, और आउटपुट सिग्नल के माध्यम से हमे सूचना दे देता है।



चित्र 7.5 : Proximity Sensor

उदाहरण—आपने देखा होगा कि जब आप फ़ोन पर बात करते हैं, यदि उस समय जब आप मोबाइल को कान के पास ले जाते है, तो फ़ोन की लाइट बन्द हो जाती है। यह सब प्रोक्सिमिटी सेंसर के उपयोग से ही होता है।

प्रोक्सिमिटी सेंसर के प्रकार (Types of Proximity Sensor)—

1. इन्डकटिव प्रोक्सिमिटी (Inductive proximity)
2. कैपसिटीव प्रोक्सिमिटी (Capacitive proximity)
3. मैग्नेटिक प्रोक्सिमिटी (Magnetic proximity)

7.2.4 प्रेशर सेंसर (Pressure Sensor)

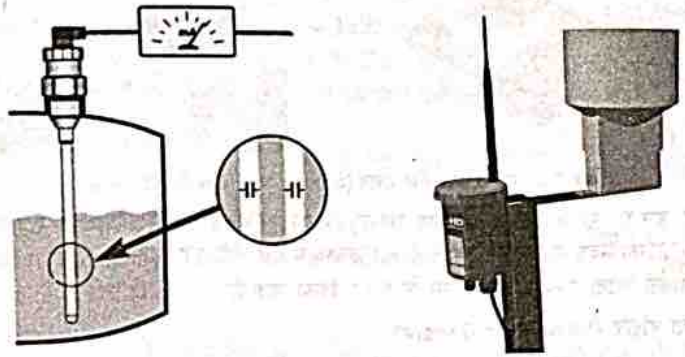
प्रेशर सेंसर का बहुत-से स्थानों पर उपयोग होता है। इस सेन्सर की मदद से किसी भी जगह के प्रेशर (दाब) का पता लगाया जा सकता है।

चित्र 7.6 : प्रेशर सेंसर (Pressure sensor)

उदहारण—आपने कभी किसी वाहन के टायर में हवा भरवाई होगी। सभी टायरों में हवा की अलग अलग जरूरत होती है, उसी के अनुसार हम उसमें हवा भरते हैं। उस समय जो सेन्सर हमारे टायर के अंदर का प्रेशर बताने का कार्य करता है, उसे ही प्रेशर सेंसर कहते हैं।

7.2.5 लेवल सेंसर (Level Sensor)

लेवल सेंसर के नाम से ही इसके उपयोग से लेवल का पता लगाया जा सकता है।



चित्र 7.7 : लेवल सेंसर (Level Sensor)

लेवल सेंसर की कार्यविधि—यदि हम पानी की टंकी को बिना देखे उसमें यह देखना चाहते हैं, कि उसमे कितना पानी भरा है, तब हमे लेवल सेंसर की आवश्यकता होती है। यह किसी भी स्तर को नापने में उपयोग किया जा सकता है।

चित्र 7.8 : लेवल सेंसर

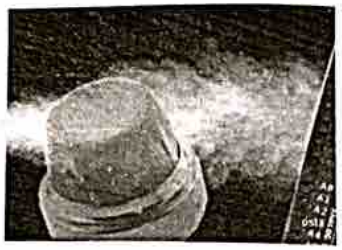
आजकल के आधुनिक वाहनों में भी लेवल सेंसर का उपयोग किया जाता है। जिनकी सहायता से हमें LED Display पर ही टंकी में पेट्रोल/डीजल की मात्रा का पता लग जाता है।

लेवल सेंसर के प्रकार (Types of Level Sensor)—

1. पॉइंट लेवल सेंसर (Point level sensor)
2. कन्टीनुअस लेवल सेंसर (Continues level sensor)

7.2.6 स्मोक और गैस सेंसर (Smoke and Gas Sensor)

गैस सेंसर को MQ2 सेंसर भी कहते हैं। यह सेंसर बहुत-सी गैसों को सेन्स करके हमे सूचना देता है। गैस सेंसर हाइड्रोजन, कार्बन, पेट्रोलियम, मीथेन, एल्कोहल, प्रोपेन आदि गैस को सेन्स कर लेते हैं।



चित्र 7.9 : स्मोक और गैस सेंसर (Smoke and Gas Sensor)

उपयोग—यदि हम घर पर हैं ओर हमको लग रहा है, कि हमारे किचन से गैस लीकेज हो रही है, तो उस समय हम स्मोक सेंसर (MQ2) की मदद से पता कर सकते है, की वास्तव में गैस लीकेज है या नही।

गैस सेंसर का सबसे ज्यादा उपयोग सेफ्टी सेंसर के रूप में किया जाता है।

7.2.7 अल्ट्रासोनिक सेंसर (Ultrasonic Sensor)

इस सेंसर का उपयोग दो कार्यों में किया जाता है।—

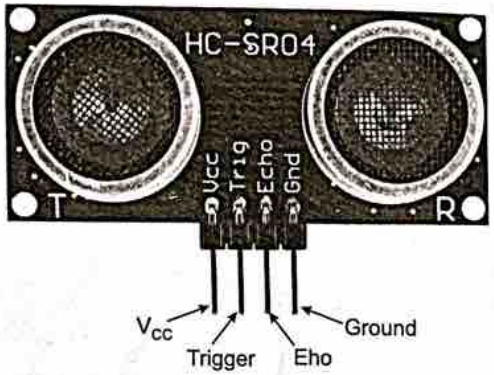
1. किसी भी वस्तु की उपस्थिति पता करने में।
2. किसी भी वस्तु की दूरी पता करने में।

अल्ट्रासोनिक सेन्सर का मुख्य उपयोग डिस्टेन्स या दूरी नापने के लिए ही किया जाता है।

इस सेन्सर के दो मुख्य भाग होते है—

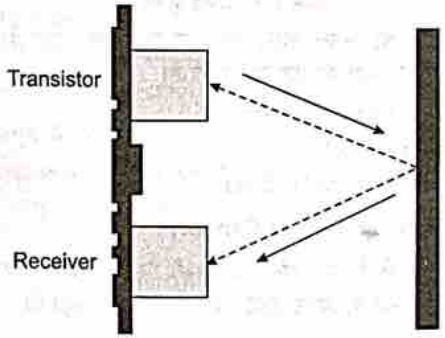
1. ट्रांसपोर्टर सेक्शन
2. रिसीवर सेक्शन

कार्यविधि—ट्रांसपोर्टर सेक्शन अपने अंदर से अल्ट्रासोनिक फ्रीक्वेंसी को बाहर भेजता है। इसके बाद यह समय की गणना करता है, कि फ्रीक्वेंसी कितने समय बाद वस्तु से टकराकर (रिफ्लेक्ट) वापस रिसीवर सेक्शन में आती है।



चित्र 7.10 : अल्ट्रासोनिक सेंसर (Ultrasonic Sensor)

इस प्रकार ultrasonic sensor किसी भी वस्तु की दूरी का पता कर पाता है।



चित्र 7.11

7.3 सेवेन सेगमेंट डिस्प्ले (Seven Segment Display)

यह डिजिटल डाटा या सूचनाओं को डिजिट या Alphanumeric रूप में प्रदर्शित करने का सबसे उपयुक्त माध्यम है। इसमें कुल 7 LED या LCD लगी होती हैं, जिन्हें सेगमेंट कहते हैं। LED में दो Point होते हैं, जिनमें से एक को Cathode एवं दूसरे को Anode कहते हैं। इनके द्वारा डेसीमल नंबर फॉर्मेट (Decimal number format) अर्थात् (0-9) एवं (A-F) को प्रदर्शित किया जाता है।

इसमें LED को अंग्रेजी के 8 की तरह करके सॉलिड रूप में रखा जाता है। सामान्यत: इन LED का रंग लाल या संतरी रखा जाता है।

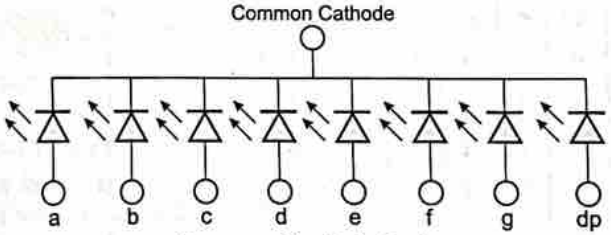
इसमें कुल 8 इनपुट कनेक्शन होते हैं, जिनमें से 1 कॉमन इनपुट टर्मिनल के लिए होता है, जबकि सात प्रत्येक LED के लिए होते हैं। इनके अतिरिक्त दशमलव के लिए एक अलग से LED एवं Point दिया होता है।

चित्र 7.12 : सेवेन सेगमेंट डिस्प्ले

7.4 सेवन सेगमेंट डिस्प्ले के प्रकार (Types of Seven Segment Display)

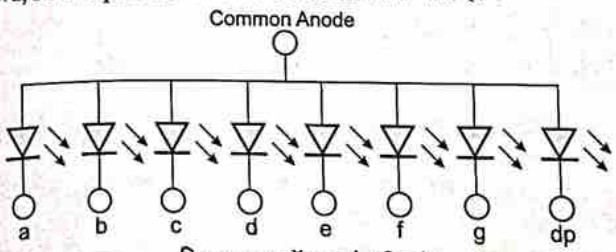
निम्नलिखित दो प्रकार के Display बाजार में उपलब्ध होते हैं—

1. कॉमन कैथोड डिस्प्ले (The Common Cathode Display (CCD))—इसे कॉमन कैथोड डिसप्ले इसलिए कहते हैं, क्योंकि इसके 8 LED के कुल 8 कनेक्शन पॉइंट में से जो कॉमन पॉइंट होता है, वह कैथोड के लिए होता है. अर्थात '0' या अर्थिंग के लिए होता है । जैसा कि चित्र में दर्शाया गया है । जबकि अन्य 7 पॉइंट्स एनोड के लिए होते है, जिसमें प्रत्येक LED के लिए अलग-अलग इनपुट दिया जाता हैं।



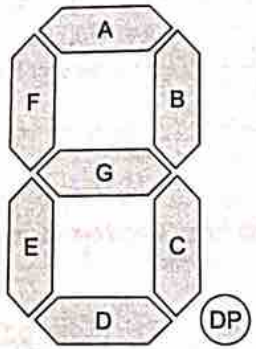
चित्र 7.13 : कॉमन कैथोड डिस्प्ले

2. कॉमन एनोड डिस्प्ले (The Common Anode Display (CAD))—यह कॉमन कैथोड डिस्प्ले का विपरीत होता है। इसमें LED के 8 कनेक्शन पॉइंट्स में से जो एक कॉमन पॉइंट होता है, वह एनोड होता है अर्थात '1' के लिए होते हैं जबकि अन्य 7 पॉइंट्स, कैथोड के लिए होता है, जिसे प्रत्येक LED के लिए अलग-अलग इनपुट दिया जाता है । इसके प्रत्येक LED को a, b c ... alphabets से निम्न प्रकार Mark किया जाता है।



चित्र 7.14 : कॉमन एनोड डिस्प्ले

चित्र में Alphabet के आधार पर प्रत्येक LED को एक पहचान दी गई है। इस आधार पर एक उदाहरण ले कि हमें 4 display कराना है, तो कौन-कौन-से LED को Glow करना चाहिए। चित्रानुसार f, g, b, c ये चार LED के जलने पर हमें digit 4 प्रदर्शित होगा।



चित्र 7.15

इसके लिए आप निम्न प्रकार की सत्य सारिणी (Truth table) भी बना सकते हैं। इस सारिणी में जिन LED की Value 1 है, उनका ऑन कंडीशन में होने की स्थिति में वह हेक्साडेसीमल नंबर या Alphabets प्रदर्शित होगा।

Segments Inputs							7 Segment Display Output
a	b	c	d	e	f	g	0
0	0	0	0	0	0	1	1
1	0	0	1	1	1	1	1
0	0	1	0	0	1	0	2
0	0	0	0	1	1	0	3
1	0	0	1	1	0	0	4
0	1	0	0	1	0	0	5
0	1	0	0	0	0	0	6
0	0	0	1	1	1	1	7
0	0	0	0	0	0	0	8
0	0	0	0	1	1	0	9

7.5 सेवेन सेगमेंट डिस्प्ले के अनुप्रयोग (Applications of Seven Segment Display)

1. Seven Segment Display का प्रयोग अधिकतर डिजिटल कैलकुलेटर, इलैक्ट्रॉनिक मीटर, डिजिटल घड़ी, ओडोमीटर, घड़ी रेडियो, आदि में होता है।
2. आज के अधिकांश 7 सेगमेंट के एप्लिकेशन एलसीडी का उपयोग कर रहे हैं, क्योंकि इनमें विद्युत की खतप बहुत कम होती है।

7.8 लिक्विड क्रिस्टल डिस्प्ले LCD (Liquid Crystal Display)

लिक्विड क्रिस्टल डिस्प्ले या LCD एक पदार्थ (Matter) की दो अवस्थाओं, ठोस और द्रव का संयोजन होता है। एलसीडी में एक विज़िबल इमेज बनाने के लिए एक तरल क्रिस्टल का उपयोग किया जाता है। लिक्विड क्रिस्टल डिस्प्ले सुपर-पतली टेक्नोलॉजी डिस्प्ले स्क्रीन होती है, जो मुख्य रूप से पर लैपटॉप कंप्यूटर स्क्रीन, टीवी, सेल फोन और पोर्टेबल वीडियो गेम में उपयोग की जाती हैं। कैथोड-रे ट्यूब (CRT) तकनीक की तुलना में LCD अधिक पतली होती हैं।

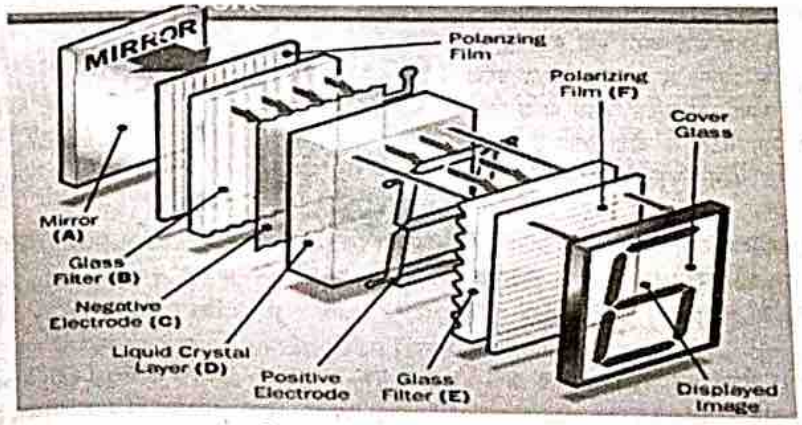
लिक्विड क्रिस्टल डिस्प्ले कई परतों से बना होता है, जिसमें दो ध्रुवीकृत पैनल फिल्टर (Polarized panel filters) और इलैक्ट्रोड होते हैं। एलसीडी तकनीक का उपयोग मिनी कम्प्यूटर; जैसे—नोटबुक या कुछ अन्य इलैक्ट्रॉनिक उपकरणों में किया जाता है। क्रिस्टल की ग्रे-स्केल छवि (Gray-scale image) के साथ रंगीन प्रकाश का संयोजन रंगीन छवि (Colored image) बनाता है। यह छवि तब स्क्रीन पर प्रदर्शित होती है।

एक एलसीडी या तो एक सक्रिय मैट्रिक्स प्रदर्शन ग्रिड (Active matrix display grid) या एक निष्क्रिय प्रदर्शन ग्रिड (A passive display grid) से बना होता है। स्मार्टफ़ोन की अधिकांश एलसीडी तकनीक में सक्रिय मैट्रिक्स डिस्प्ले का उपयोग किया जाता है, लेकिन कुछ पुराने डिस्प्ले अभी भी निष्क्रिय डिस्प्ले ग्रिड डिज़ाइन का उपयोग करते हैं। अधिकांश इलैक्ट्रॉनिक उपकरण मुख्य रूप से लिक्विड क्रिस्टल डिस्प्ले तकनीक पर आधारित होते हैं। लिक्विड में एलईडी या कैथोड-रे ट्यूब की तुलना में कम बिजली की खपत होने का एक अलग लाभ है। लिक्विड क्रिस्टल डिस्प्ले स्क्रीन प्रकाश उत्सर्जित (Emitting light) करने के स्थान पर प्रकाश को अवरुद्ध करने के सिद्धान्त (Principle of blocking light) पर कार्य करती है। एलसीडी को बैकलाइट की आवश्यकता होती है, क्योंकि यह प्रकाश का उत्सर्जन नहीं करती। हम आजकल उन उपकरणों का उपयोग करते हैं, जिनमें LCD Display होता है कैथोड-रे ट्यूब एलसीडी की तुलना में अधिक पॉवर लेती है तथा भारी और बड़ी भी होती है।

7.9 LCD की सरंचना (Construction of LCD)

LCD बनाते समय सरल तथ्यों पर विचार किया जाना चाहिए—

1. LCD की मूल संरचना को एप्लाइड करंट (Applied current) में बदलकर नियंत्रित किया जाना चाहिए।
2. हमें ध्रुवीकृत प्रकाश (Polarized light) का उपयोग करना चाहिए।
3. लिक्विड क्रिस्टल को संचरित करने के लिए दोनों ऑपरेशन को नियंत्रित करने में सक्षम होना चाहिए या ध्रुवीकृत प्रकाश को बदलने में भी सक्षम होना चाहिए।



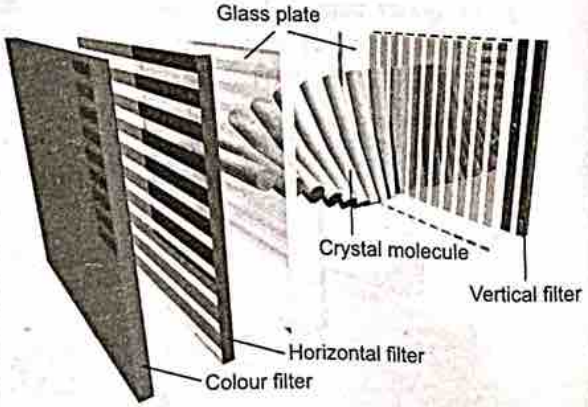
चित्र 7.16 : LCD Construction

जैसा कि ऊपर उल्लेख किया गया है, कि हमें लिक्विड क्रिस्टल बनाने में दो ध्रुवीकृत काँच के टुकड़ों की आवश्यकता होती है। जिस ग्लास की सतह पर ध्रुवीकृत फिल्म नहीं होती है, उसे एक विशेष पॉलीमर से रगड़ना चाहिए, जो ध्रुवीकृत ग्लास फिल्टर की सतह पर माइक्रोस्कोपिक ग्रूवस (Microscopic grooves) बनाएगा। ग्रूवस को ध्रुवीकृत फिल्म के समान दिशा में होना चाहिए। अब हमें ध्रुवीकृत काँच के ध्रुवीकरण फिल्टर में से एक पर वायवीय लिक्विड फेज क्रिस्टल (Pneumatic liquid phase crystal) की एक परत को जोड़ना होगा। माइक्रोस्कोपिक चैनल फिल्टर अभिविन्यास (Filter orientation) के साथ संरेखित करने के लिए पहली परत अणु का कारण बनता है। जब पहली परत के टुकड़े पर समकोण दिखाई देता है, तो हमें ध्रुवीकृत फिल्म के साथ काँच का दूसरा टुकड़ा जोड़ना चाहिए। पहला फिल्टर स्वाभाविक रूप से ध्रुवीकृत होगा, क्योंकि प्रारंभिक चरण में प्रकाश इससे स्ट्राइक करता है। इस प्रकार प्रकाश प्रत्येक परत के माध्यम से यात्रा करता है और एक अणु की मदद से अगले तक निर्देशित होता है। अणु अपने कोण से मेल खाने के लिए प्रकाश के कंपन के अपने प्लेन को बदलने के लिए जाता है। जब प्रकाश लिक्विड क्रिस्टल पदार्थ के दूर के छोर तक पहुँचता है, तो यह उसी कोण पर कंपन करता है, जो अणु के कम्पन की अंतिम परत से होता है। प्रकाश को डिवाइस में प्रवेश करने की अनुमति केवल तभी होती है, जब ध्रुवीकृत काँच की दूसरी परत अणु की अंतिम परत के साथ मेल खाती है।

7.10 LCD के कार्यविधि (Working)

LCD का सिद्धान्त यह है, कि जब एक विद्युत धारा को तरल क्रिस्टल अणु पर लागू किया जाता है, तो अणु अछूता हो जाता है। यह प्रकाश के कोण का कारण बनता है जो ध्रुवीकृत काँच के अणु से गुजर रहा है और शीर्ष ध्रुवीय फिल्टर के कोण में भी परिवर्तन का कारण बनता है। परिणामस्वरूप LCD के एक विशेष क्षेत्र के माध्यम से ध्रुवीकृत ग्लास को पारित करने के लिए थोड़ी रोशनी पास होती है। इस प्रकार वह विशेष क्षेत्र अन्य की तुलना में काला हो जाएगा। LCD प्रकाश को अवरुद्ध करने के सिद्धांत पर कार्य करती है। LCD का निर्माण करते समय एक प्रतिबिंबित दर्पण को पीछे की ओर लगाया जाता है। इलेक्ट्रोड प्लेन इंडियम-टिन-ऑक्साइड से बना होता है जिसे ऊपर रखा जाता है और ध्रुवीकृत फिल्म के साथ एक ध्रुवीकृत ग्लास भी डिवाइस के तल पर जोड़ा जाता है। LCD का पूरा क्षेत्र एक इलेक्ट्रोड द्वारा संलग्न किया जाना है और इसके ऊपर तरल क्रिस्टल पदार्थ होता है ।

इसके बाद नीचे की ओर आयत के रूप में एक इलैक्ट्रोड के साथ काँच का दूसरा टुकड़ा आता है, और ऊपर, एक और ध्रुवीकरण फिल्म। यह माना जाना चाहिए कि दोनों टुकड़े सही कोण पर रखे गए हैं। जब कोई करंट नहीं होता है, तो प्रकाश LCD के सामने से गुजरता है यह दर्पण द्वारा परिलक्षित होगा और वापस बाउंस होगा। जैसा कि इलैक्ट्रोड एक बैटरी से जोड़ा जाता है, इससे होने वाला करंट कॉमन-प्लेन इलैक्ट्रोड के बीच लिक्विड क्रिस्टल का कारण बनता है और इलैक्ट्रोड को आयताकार की तरह आकार देता है। इस प्रकार प्रकाश को गुजरने से रोक दिया जाता है। वह विशेष रूप से आयत क्षेत्र खाली दिखाई देता है।



चित्र 7.17

एक LCD पैनल कई परतों से बना होता है। इनमें एक पोलराइज़र, ध्रुवीकृत ग्लास, LCD द्रव, प्रवाहकीय कनेक्शन आदि होते हैं। (An LCD panel is made of many layers. These consist of a polarizer, polarized glass, LCD fluid, conductive connections etc.)

ध्रुवीकरण एक ऐसी प्रक्रिया है, जिसमें प्रकाश तरंगों का कंपन एक प्लेन तक सीमित होता है, जिसके परिणामस्वरूप प्रकाश तरंगों का निर्माण ध्रुवीकृत प्रकाश के रूप में होता है। (Polarization is a process in which the vibration of light waves is restricted to a single plane, resulting in the formation of light waves known as polarized light.)

चूंकि लिक्विड क्रिस्टल का स्वयं का प्रकाश नहीं होता है, उन्हें कार्य करने के लिए बाहरी प्रकाश स्रोत की आवश्यकता होती है। LCD पैनल में ध्रुवीकृत काँच के सेट होते हैं, जिनके बीच में लिक्विड क्रिस्टल सामग्री होती है। जब बाहरी प्रकाश एक ध्रुवीकृत ग्लास से गुजरता है और तरल क्रिस्टल अणुओं पर विद्युत धारा लागू होती है, तो वे स्वयं को इस तरह से संरेखित करते हैं, कि ध्रुवीकृत प्रकाश पहली परत से दूसरी ध्रुवीकृत काँच तक यात्रा करता है, जिससे स्क्रीन पर एक छवि दिखाई देती है।

7.11 LCD के प्रकार (Types of LCD)

7.11.1 ट्विस्टेड नेमैटिक डिसप्ले (Twisted Nematic Display)

TN (ट्विस्टेड नेमैटिक) LCDs का उत्पादन आसानी से किया जा सकता है । ये सबसे अधिक गेमर्स द्वारा उपयोग किए जाते हैं, क्योंकि ये अन्य डिस्प्ले की तुलना में सस्ते और Quick response time वाले होते हैं। इन डिस्प्ले का मुख्य नुकसान यह है, कि इनमें कम गुणवत्ता के साथ-साथ आंशिक कंट्रास्ट अनुपात (Partial contrast ratio), देखने के कोण (Viewing angles) और कलर का रिप्रोडक्शन होता है। लेकिन, ये उपकरण दैनिक संचालन के लिए पर्याप्त हैं। ये एकमात्र गेमिंग डिस्प्ले हैं, जो 240 हट्र्ज में उपलब्ध हैं। इन डिस्प्ले में खराब कंट्रास्ट और कलर होता है।

7.11.2 इन-प्लेन स्विचिंग डिसप्ले (In-Plane Switching Display)

IPS डिस्प्ले को सबसे अच्छा LCD माना जाता है, क्योंकि ये अच्छी इमेज गुणवत्ता, उच्च देखने के कोण, वाइब्रेंट कलर प्रिसिशन और डीफ्रेंस प्रदान करते हैं। ये डिस्प्ले अधिकतर ग्राफिक डिजाइनर और कुछ अन्य अनुप्रयोगों में उपयोग किए जाते हैं, एलसीडी को इमेज और कलर के रिप्रोडक्शन के लिए अधिकतम संभावित मानकों की आवश्यकता होती है।

7.11.3 वर्टीकल एलाइनमेंट पैनल (Vertical Alignment Panel)

वर्टिकल अलाइनमेंट (VA) पैनल ट्विस्टेड नेमैटिक और इन-प्लेन स्विचिंग पैनल टेक्नोलॉजी के बीच में आते हैं। इन पैनलों में TN प्रकार के डिस्प्ले की तुलना में उच्च गुणवत्ता वाले फीचर्स के साथ बेहतरीन Viewing angle और कलर रिप्रोडक्शन होता है। इन पैनलों का कम प्रतिक्रिया समय होता है। ये दैनिक उपयोग के लिए बहुत अधिक उचित और उपयुक्त होता है।

इस पैनल की संरचना ट्विस्टेड नेमेटिक डिसप्ले की तुलना में गहरे काले रंग के साथ-साथ बेहतर रंगों के डिस्प्ले उत्पन्न करती है। और कई क्रिस्टल संरेखण (Crystal alignments) TN प्रकार के डिस्प्ले की तुलना में बेहतर Viewing angle देते हैं। ये डिस्प्ले एक ट्रेडऑफ़ के साथ आते हैं, क्योंकि ये अन्य डिस्प्ले की तुलना में महंगे होते हैं तथा इनमें धीमी प्रतिक्रिया समय और कम ताज़ा दरें होती हैं।

7.11.4 एडवांस्ड फ्रिंज फील्ड स्विचिंग LCD (Advanced Fringe Field Switching (AFFS))

AFFS LCDs , IPS डिस्प्ले की तुलना में सबसे अच्छी परफॉरमेंस और कलर रिप्रोडक्शन की एक बड़ी रेंज प्रदान करता है। AFFS के अनुप्रयोग बहुत उन्नत हैं, क्योंकि ये Viewing angle पर समझौता किए बिना कलरडिसटोरशन को कम कर सकते हैं। आमतौर पर, इस डिसप्ले का उपयोग उच्च आधुनिक और व्यावसायिक परिवेश जैसे—व्यवहार्य हवाई-जहाज के कॉकपिट (Viable airplane cockpits) में किया जाता है।

7.11.5 पैसिव और एक्टिव मैट्रिक्स डिसप्ले (Passive and Active Matrix Displays)

पैसिव-मैट्रिक्स टाइप LCD एक साधारण ग्रिड के साथ कार्य करता है, ताकि LCD पर एक विशिष्ट पिक्सल तक चार्ज किया जा सके। ग्रिड को एक शांत प्रक्रिया के साथ डिजाइन किया जाता है और यह दो सबस्ट्रेट्स के माध्यम से शुरू होता है, जो कि काँच की परत होती है। एक ग्लास लेयर कॉलम देती है, जबकि दूसरी पंक्तियाँ देती है जो इंडियम-टिन-ऑक्साइड जैसी स्पष्ट चालक सामग्री का उपयोग करके डिज़ाइन की जाती हैं।

जब भी चार्ज किसी विशेष पंक्ति या कॉलम की दिशा में प्रेषित होता है, तो इस डिस्प्ले में पंक्तियों अन्यथा कॉलम को ICs से जोड़ा जाता है। लिक्विड क्रिस्टल के मटेरियल को काँच की दो परतों के बीच रखा जाता है, जहाँ सब्सट्रेट के बाहरी तरफ, एक ध्रुवीकरण फिल्म को जोड़ा जा सकता है। IC एक सब्सट्रेट के कॉलम के नीचे एक चार्ज को प्रसारित करता है ।

सक्रिय-मैट्रिक्स प्रकार के LCD मुख्य रूप से TFT (Thin Film Transistor) पर निर्भर करते हैं। ये ट्रांजिस्टर छोटे स्विचिंग ट्रांजिस्टर के साथ-साथ कैपेसिटर होते हैं, जो एक ग्लास सब्सट्रेट पर एक मैट्रिक्स के भीतर रखे जाते हैं। जब उचित पंक्ति सक्रिय हो जाती है, तो एक चार्ज को सटीक कॉलम के नीचे प्रेषित किया जा सकता है, ताकि एक विशिष्ट पिक्सल को सम्बोधित (addressed) किया जा सके, क्योंकि स्तम्भ के अतिरिक्त सभी पंक्तियों को बंद कर दिया जाता है, बस निर्दिष्ट पिक्सल में संधारित्र को चार्ज मिलता है।

7.12 LCD के लाभ (Advantages of LDC)

LCD के निम्नलिखित लाभ हैं—

- LCD में CRT और LED की तुलना में कम मात्रा में बिजली की खपत होती है।
- LED की तुलना में डिसप्ले के लिए LCD में कुछ माइक्रोवाट्स होते हैं।
- LCD कम लागत वाली होती हैं।
- उत्कृष्ट कंट्रास्ट प्रदान करती है।
- कैथोड-रे ट्यूब और LED की तुलना में LCD पतली और हल्की होती है।

7.13 LCD के नुकसान (Disadvantaes of LCD)

लिक्विड क्रिस्टल डिस्प्ले के नुकसान निम्नलिखित हैं—

- अतिरिक्त प्रकाश स्रोतों की आवश्यकता होती है।
- ऑपरेशन के लिए टेम्परेचर रेंज सीमित है।
- कम विश्वसनीयता होती है।
- गति बहुत कम होती है।
- LCD को AC ड्राइव की आवश्यकता होती है।

7.14 LCD के अनुप्रयोग (Applications of LCD)

LCD के निम्नलिखित अनुप्रयोग हैं—

- तकनीक में विज्ञान और इंजीनियरिंग के साथ-साथ इलैक्ट्रॉनिक उपकरणों के क्षेत्र में प्रमुख अनुप्रयोग हैं।
- लिक्विड क्रिस्टल थर्मामीटर।
- ऑप्टिकल इमेजिंग।
- LCD तकनीक वेवगाइड में रेडियो आवृत्ति तरंगों के दृश्य में भी लागू होती है।
- चिकित्सा अनुप्रयोगों में उपयोग किया जाता है।

7.15 लाइट ईमिटिंग डायोड (Light Emitting Diode (LED))

LED का पूरा नाम Light Emitting Diode है। यह आज उपलब्ध सभी विभिन्न प्रकार के सेमीकंडक्टर डायोड में से सबसे अधिक उपयोग किया जाने वाला सेमीकंडक्टर डायोड हैं। LED फोरवोर्ड बायस होने पर दृश्यमान प्रकाश (Visible light) या अदृश्य इंफ्रारेड प्रकाश (Invisible infrared light) का उत्सर्जन करते हैं। LED जो Invisible infrared light उत्सर्जित करते हैं, उनका उपयोग रिमोट कंट्रोल में किया जाता है।

LED एक ऑप्टिकल सेमीकंडक्टर डिवाइस है, जो वोल्टेज लागू होने पर प्रकाश का उत्सर्जन करता है। अन्य शब्दों में, LED एक ऑप्टिकल सेमीकंडक्टर डिवाइस है, जो विद्युत ऊर्जा को प्रकाश ऊर्जा में परिवर्तित करता है।

चित्र 7.18 : एल०ई०डी०

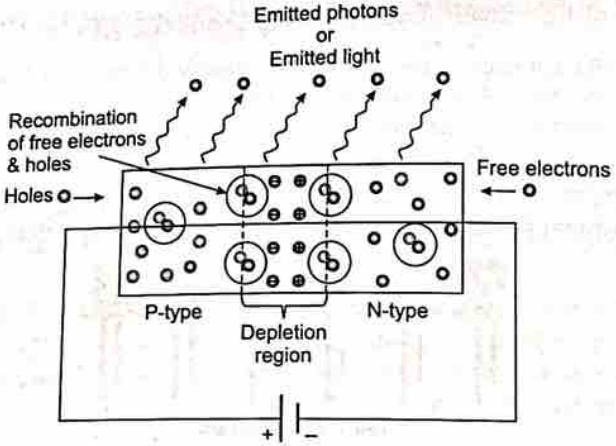
जब LED फॉरवर्ड बायस्ड होता है, तो कंडक्शन बैंड में मुक्त इलैक्ट्रॉन्स वेलेंस बैंड में होल्स के साथ रिकंबाइन होते है और प्रकाश के रूप में ऊर्जा जारी करते है। विद्युत क्षेत्र या इलैक्ट्रिक करंट के प्रवाह के जवाब में प्रकाश उत्सर्जित करने की प्रक्रिया को इलैक्ट्रोल्यूमिनेशन (Electroluminescence) कहते हैं।

एक सामान्य *p-n* जंक्शन डायोड केवल एक दिशा में इलैक्ट्रिक करंट की अनुमति देता है। यह Forward Biased होने पर विद्युत धारा की अनुमति देता है और Revirse Biased होने पर इलैक्ट्रिक करंट की अनुमति नहीं देता। इस प्रकार, सामान्य *p-n* जंक्शन डायोड केवल Forward Biased Condition में ऑपरेट होता है। सामान्य *p-n* जंक्शन डायोड की तरह, LED भी केवल Forward Biased Condition में ऑपरेट होता है। एक LED बनाने के लिए, *n*-टाइप मटेरियल को बैटरी के निगेटिव टर्मिनल से जोड़ा जाना चाहिए और *p*-टाइप मटेरियल को बैटरी के पॉजिटिव टर्मिनल से जोड़ा जाना चाहिए। दूसरे शब्दों में, *n*-type मटेरियल को निगेटिव रूप से चार्ज किया जाना चाहिए और *p*-type मटेरियल को पॉजिटिव रूप से चार्ज किया जाना चाहिए।

LED का निर्माण सामान्य *p-n* जंक्शन डायोड के समान है। इसमें निर्माण के लिए सिलिकॉन या जर्मेनियम के स्थान पर गैलियम, फास्फोरस और आर्सेनिक मटेरियल का उपयोग किया जाता है। सामान्य *p-n* जंक्शन डायोड में सिलिकॉन का व्यापक रूप से उपयोग किया जाता है, क्योंकि यह तापमान के प्रति कम संवेदनशील होता है। इसके अतिरिक्त, यह बिना किसी नुकसान के कुशलतापूर्वक विद्युत धारा का प्रवाह करता है। कुछ मामलों में जर्मेनियम का उपयोग डायोड के निर्माण के लिए किया जाता है। हालांकि, सिलिकॉन या जर्मेनियम डायोड प्रकाश के रूप में ऊर्जा का उत्सर्जन नहीं करते हैं। इसके विपरीत, वे ऊष्मा के रूप में ऊर्जा को उत्सर्जित करते हैं। इस प्रकार, LED के निर्माण के लिए सिलिकॉन या जर्मेनियम का उपयोग नहीं किया जाता है।

7.16 LED की कार्यविधि (Working of LED)

LED केवल Forward Biased Condition में काम करता है। जब LED Forward Biased हो जाता है, तो *n*-साइड से मुक्त इलैक्ट्रॉन्स और *p*-साइड से होल्स को जंक्शन की ओर धकेल दिया जाता है। जब मुक्त इलेक्ट्रॉन जंक्शन या डिपलीसन क्षेत्र में पहुँचते हैं, तो कुछ मुक्त इलैक्ट्रॉन धनात्मक आयनों के होल्स् से पुन: जुड़ जाते हैं। हम जानते हैं, कि पॉजिटिव आयनों में प्रोटॉन की तुलना में इलैक्ट्रॉन की संख्या कम होती है। इसलिए, वे इलैक्ट्रॉन को स्वीकार कर लेते हैं। इस प्रकार, मुक्त इलैक्ट्रॉन डिपलीसन क्षेत्र में होल्स के साथ रि-कम्बाइन होते हैं। डिपलीसन क्षेत्र में मुक्त इलैक्ट्रॉन और होल्स के रि-कॉम्बिनेशन के कारण डिपलीसन क्षेत्र की चौड़ाई कम हो जाती है। परिणामस्वरूप, अधिक चार्ज वाहक *p-n* जंक्शन को पार करेंगे।



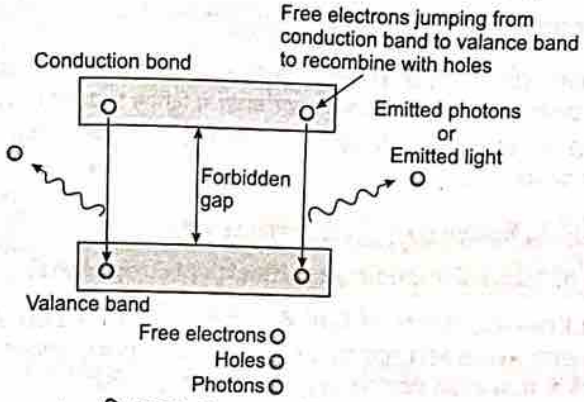
चित्र 7.19 : Light Emitting Diode (LED)

p-साइड और n-साइड से कुछ चार्ज वाहक p-n जंक्शन को पार कर जाते हैं, इससे पहले कि वे डिपलीसन क्षेत्र में रि-कम्बाइन हो। उदाहरण के लिए, n-टाइप सेमीकंडक्टर से कुछ मुक्त इलेक्ट्रॉन्स p-n जंक्शन को पार करते हैं और p-टाइप प्रकार के में होल्स के साथ रि-कम्बाइन होते हैं। इसी प्रकार, p-टाइप सेमीकंडक्टर से होल्स p-n जंक्शन को पार करते हैं और n-टाइप सेमीकंडक्टर में मुक्त इलैक्ट्रॉनों के साथ रि-कम्बाइन होते हैं। इस प्रकार, रि-कॉम्बिनेशन डिपलीसन n क्षेत्र के साथ-साथ p-टाइप और n-टाइप सेमीकंडक्टर में होता है। कंडक्टसन बैंड में मुक्त इलैक्ट्रॉन को वेलेंस बैंड में होल्स के साथ रि-कम्बाइन होने से पहले प्रकाश के रूप में ऊर्जा जारी होती है।

7.17 LED से प्रकाश का उत्सर्जन (Emitting of Light From LED)

जब बाह्य वोल्टेज को वेलेंस इलैक्ट्रॉनों पर लागू किया जाता है, तो वे पर्याप्त ऊर्जा प्राप्त करते हैं और मूल परमाणु के साथ बॉन्डिंग तोड़ते हैं। जो वेलेंस इलैक्ट्रॉन मूल परमाणु के साथ बॉन्डिंग तोड़ते हैं, उन्हें मुक्त इलैक्ट्रॉन कहते हैं। जब वेलेंस इलैक्ट्रॉन अपने पैरेंट परमाणु को छोड़ देते हैं, तो वे वेलेंस शेल में एक खाली स्थान छोड़ देते हैं। वेलेंस शेल के इस खाली स्थान को होल कहते हैं। सभी वेलेंस इलैक्ट्रॉनों की ऊर्जा का स्तर लगभग समान होता है। सभी वेलेंस इलैक्ट्रॉनों की ऊर्जा के स्तर के ग्रुप को वेलेंस बैंड कहते हैं। इसी तरह, सभी मुक्त इलैक्ट्रॉनों की ऊर्जा का स्तर लगभग समान होता है। सभी मुक्त इलैक्ट्रॉनों की ऊर्जा स्तर के क्षेत्र में समूह को Conduction Band कहते हैं।

Conduction Band मुक्त इलैक्ट्रॉन की ऊर्जा का स्तर वेलेंस इलैक्ट्रॉनों की ऊर्जा के स्तर या वेलेंस बैंड में होल्स की तुलना में अधिक होता है। इसलिए, कंडक्सन बैंड में मुक्त इलैक्ट्रॉनों को वेलेंस बैंड के होल्स के साथ रिकंबाइन करने में नष्ट हुई ऊर्जा की आवश्यकता होती है। कंडक्शन बैंड में मुक्त इलैक्ट्रॉन लम्बे समय तक नहीं रहते। थोड़े समय के बाद, मुक्त इलैक्ट्रॉन लाइट के रूप में ऊर्जा को नष्ट कर देते हैं और वेलेंस बैंड में होल्स के साथ रिकंबाइन होते हैं। चार्ज वाहक का प्रत्येक रिकॉम्बिनेशन कुछ प्रकाश ऊर्जा का उत्सर्जन करता है।



चित्र 7.20 : Process of Emission in LED

मुक्त इलैक्ट्रॉन से नष्ट हुई ऊर्जा या उत्सर्जित प्रकाश की तीव्रता, Conduction Band एवं Balance Band के बीच फोर्बिडन गैप या एनर्जी गैप पर निर्भर करती है। बड़े फोर्बिडन गैप वाले सेमीकंडक्टर डिवाइस उच्च तीव्रता के प्रकाश का उत्सर्जन करते हैं जबकि छोटे फोर्बिडन गैप वाले सेमीकंडक्टर डिवाइस कम तीव्रता के प्रकाश का उत्सर्जन करते हैं। अन्य शब्दों में, उत्सर्जित प्रकाश की ब्राइटनेस LED के निर्माण में उपयोग किए जाने वाले मटेरियल पर निर्भर करती है।

सामान्य सिलिकॉन डायोड में Conduction Band और Balance Band के बीच ऊर्जा का अंतर कम होता है। इसलिए, इलैक्ट्रॉन केवल थोड़ी दूरी पर गिरते हैं। परिणामस्वरूप, कम ऊर्जा के फोटॉन उत्पन्न होते हैं। इन कम ऊर्जा वाले फोटॉन की फ्रीक्वेंसी कम होती है, जो मानव की आंखों के लिए अदृश्य होती है। LED में, रिकॉम्बिनेशन प्रोसेस के कारण प्रकाश उत्पन्न होता है। चार्ज वाहक के रिकॉम्बिनेशन केवल Forward Bias Condition कंडिशन में होते है। इसलिए, LED केवल Forward Bias की स्थिति में काम करती हैं।

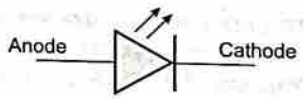
जब LED Reverse Bias होता है, तो n-साइड से मुक्त इलैक्ट्रॉन्स (वाहक) और p-साइड से होल्स जंक्शन से दूर चले जाते हैं। परिणामस्वरूप, डिपलिसन क्षेत्र की चौड़ाई बढ़ जाती है और चार्ज कैरियर्स का रिकॉम्बिनेशन नहीं होता। इस प्रकार, कोई प्रकाश उत्पन्न नहीं होता है। यदि LED पर लागू किया गया Reverse Bias वोल्टेज अत्यधिक बढ़ जाता है, तो डिवाइस को नुकसान भी हो सकता है।

सभी डायोड फोटॉन या प्रकाश का उत्सर्जन करते हैं, लेकिन सभी डायोड Visible Light का उत्सर्जन नहीं करते। एक LED में मटेरियल इस प्रकार चुना जाता है कि जारी किए गए फोटॉनों को वेवलेंथ लाइट स्पेक्ट्रम के Visible Part के भीतर आती है।

LED को 1ns की बहुत तेज गति से ऑन और ऑफ किया जा सकता है।

7.18 LED का संकेत चिह्न (Light Emitting Diode (LED) Symbol)

LED का संकेत चिह्न सामान्य p-n जंक्शन डायोड के समान है, सिवाय इसके कि इसमें डायोड से दूर इंडिकेट करने वाले एरो होते हैं जो यह निर्दिष्ट करते हैं, कि डायोड द्वारा प्रकाश उत्सर्जित किया जा रहा है।



चित्र 7.21 : LED symbol

7.19 LED के प्रकार (Types of LED)

आजकल इलैक्ट्रॉनिक्स उद्योग में LED का सबसे अधिक उपयोग किया जाता है। यह विभिन्न आकारों और आकृतियों में उपलब्ध है। उपयोगकर्ता की आवश्यकता के अनुसार बाजार में विभिन्न प्रकार के LED लाइट उपलब्ध हैं।

मुख्य रूप से LED को उनके विद्युत गुणों और उनके निर्माण के लिए उपयोग किए जाने वाले मटेरियल के आधार पर विभाजित किया जा सकता है।

7.20 विद्युत गुणों के आधार पर LED के मुख्य प्रकार (Types of LEDs Depending on Electrical Properties)

1. AC Driven LEDs—DC कनवर्टर की किसी भी आवश्यकता के बिना ये LED, AC पॉवर पर कार्य कर सकते हैं। Seoul सेमीकंडक्टर Acrich MJT नाम का एक उच्च वोल्टेज LED एक साधारण कंट्रोल सर्किट के साथ AC पॉवर से ड्राइविंग करने में सक्षम है। इस प्रकार की LED की दक्षता 40 lm/W होती है।

2. Miniature LED—छोटे आकार के LED का उपयोग अधिकतर इन दिनों इसकी परफॉर्मेंस स्पीड और अच्छी एफिशिएंसी के कारण किया जाता है। ऑप्टिकल कम्यूनिकेशन के लिए कुशल बिजली, नैनो लेजर्स शोधकर्ताओं ने 2D फ्लेक्सिबल मटेरियल से बने सबसे पतले LED का आविष्कार किया है, जो 3D LED की तुलना में 10 से 20 गुना पतला है। मिनिएचर सिंगल डाइ LED कम करंट आमतौर पर 2 mA के होते हैं।

3. High Power LEDs—इस प्रकार के LED को सैकड़ों mA से एक एम्पीयर से अधिक करेंट में संचालित किया जा सकता है। हाई पॉवर LED में हीट डीसीपेसन के लिए हिट सिंक रखा जाता है, क्योंकि यदि Hp-LED से गर्मी को हटाया नहीं जाता है, तो यह डिवाइस को नुकसान पहुँचा सकता है। इसे आसानी से एक शक्तिशाली LED लैंप बनाने के लिए एक सरणी में सेट किया जा सकता है।

7.21 मटेरियल के आधार पर LED के प्रकार (Types of LEDs Depending on the Material)

Light Emitting Diodes सेमीकंडक्टर कम्पाउंड यानी इसके हल्के रंग और आगे के बायस्ड LED करंट के आधार पर करंट डिपेंडेंट डिवाइस होते हैं। विभिन्न प्रकार के सेमीकंडक्टर, धातु और गैस कंपाउंड को मिलाकर बड़ी संख्या में LED प्राप्त होते हैं। उनमें से कुछ नीचे दिए गए हैं—

- Zinc selenide (*ZnSe*)
- Gallium Nitride (*GaN*)
- Gallium Phosphide (*GaP*)
- Silicon Carbide (*SiC*)
- Gallium Arsenide (*Ga As*)
- Gallium Arsenide Phosphide (*Ga AsP*)

7.22 LED के लाभ (Advantages of LED)

LED द्वारा उत्सर्जित प्रकाश की चमक LED के माध्यम से बहने वाले करंट पर निर्भर करती है। इसलिए, LED की चमक को करंट में परिवर्तित करके आसानी से नियंत्रित किया जा सकता है। यह विभिन्न परिवेश प्रकाश व्यवस्था की परिस्थितियों में LED डिस्प्ले ऑपरेट करना संभव बनाता है।

इस प्रकार LED के लाभ निम्नलिखित हैं—

- Light Emitting Diodes कम ऊर्जा का उपभोग करते हैं।
- LED बहुत सस्ते और आसानी से उपलब्ध हैं।
- LED वजन में हल्के होते हैं।
- आकर में छोटे होते हैं।
- LED का जीवनकाल अधिक होता है।
- यह बहुत तेजी से संचालित होता है। इन्हें बहुत कम समय में ऑन और ऑफ किया जा सकता है।
- इनमें पारा जैसा विषाक्त पदार्थ नहीं होता जैसे फ्लोरोसेंट लैंप में उपयोग किया जाता है।
- ये प्रकाश के विभिन्न रंगों का उत्सर्जन कर सकते हैं।

7.23 LED से हानियाँ (Disadvantages of LED)

सामान्य *p-n* जंक्शन डायोड की तुलना में LED को संचालित करने के लिए अधिक पॉवर की आवश्यकता होती है। LED की चमकदार दक्षता कम होती है।

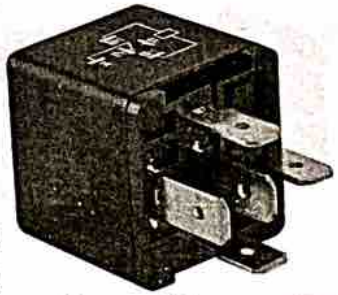
7.24 LED के अनुप्रयोग (Applications of LED)

LED के विभिन्न अनुप्रयोग इस प्रकार हैं—

- बर्गलर अलार्म सिस्टम
- चित्र फोन
- डिजिटल कंप्यूटर
- माइक्रोप्रोसेसर
- ऑटोमोटिव हीट लैंप
- विमान की लाइट
- कैलकुलेटर
- ट्रैफिक सिग्नल
- मल्टीमीटर
- डिजिटल घड़ी
- कैमरे की फ्लैश

7.25 रिले (Relay)

रिले एक ऐसा इलैक्ट्रॉनिक स्विच होता है, जिसकी मदद से हम किसी भी सर्किट को ON/OFF कर सकते हैं। रिले में एक विद्युत चुम्बक होता है, जो कि किसी पॉवर सोर्स से जुड़ा होता है तथा विद्युत चुम्बक की सहायता से किसी अन्य एक या एक से अधिक सर्किट को ऑन या ऑफ किया जा सकता है उदाहरण – थर्मल ओवरलोड रिले, इस रिले का उपयोग AC मोटर्स की सुरक्षा के लिए किया जाता है। जब मोटर ओवरलोड हो जाती है तो रिले मोटर की सप्लाई को डिसकनेक्ट कर देती है और मोटर जलने से बच जाती है।



चित्र 7.22 : रिले (Relay)

रिले लो वोल्टेज पर चलने वाला स्विच है, जिससे सप्लाई को ON या OFF कर सकते हैं घरों में सामान्य स्विच लगाए जाते हैं, जिन्हें हम हाथ से ON या OFF करते हैं, लेकिन रिले का उपयोग करने के लिए रिले को सप्लाई देनी पड़ती है, तभी वह अपना कार्य करती है, जहाँ पर भी स्विचिंग का कार्य होता है, जैसे कि UPS, स्टेबलाईजर इत्यादि में रिले का उपयोग अवश्य किया जाता है। आजकल स्विचिंग के लिए सेमीकंडक्टर डिवाइस का उपयोग किया जाता है परन्तु आज भी बहुत-से सर्किट में रिले का उपयोग किया जाता है।

रिले का कार्य करने का तरीका बहुत आसान है। रिले में सामान्यतः एक Coil (कुंडली) लगी होती है, जो इसमें लगे नार्मल कांटेक्ट (NC) को नार्मल ओपन (NO) में बदल देती है, जब रिले ऑफ होगी तब कॉमन टर्मिनल सीधा NC कांटेक्ट से जुड़ जाता है। आपने कॉमन टर्मिनल पर कोई सप्लाई दी, तो वह सीधे NC कांटेक्ट पर जाएगी परन्तु जैसे ही हम रिले की कुंडली को सप्लाई देंगे, तो यह कुंडली सक्रिय हो जाती है और आर्मेचर को अपनी ओर खींच लेती है, जिससे कि कॉमन टर्मिनल अब NC कांटेक्ट से हटकर NO कांटेक्ट से जुड़ जाएगा, लेकिन जैसे ही रिले की सप्लाई बंद करेंगे तब इसमें लगा स्प्रिंग वापिस आर्मेचर को अपनी ओर खींच लेगा और फिर से कॉमन टर्मिनल NC कांटेक्ट से जुड़ (Connect) जाता है

रिले को फ़ास्ट स्विचिंग करनें के लिए उसके सभी पार्ट्स का सही से कार्य करना आवश्यक है। अब हम रिले के भागों के बारें में जानते हैं। रिले के पाँच भाग (Components or elements) होते हैं—

1. **Coil**—रिले की Coil को जब सप्लाई देते हैं, तब वह आर्मेचर को अपनी ओर खींचती हैं। और NC को NO में बदल देती है।
2. **Yoke**—रिले के बाहरी प्लास्टिक भाग को योक कहते हैं।
3. **Contact**—स्विचिंग के समय NO, NC के रूप में कांटेक्ट का प्रयोग किया जाता है।
4. **Armature**—रिले के अंदर कॉमन टर्मिनल से जुड़ा होता है।
5. **Spring**—रिले के अंदर स्प्रिंग का अपना अलग स्थान होता है। जब रिले की कुंडली को सप्लाई मिलती है तब आर्मेचर NO से जुड़ जाता है तथा कुंडली की सप्लाई काटने के बाद स्प्रिंग आर्मेचर को खींचकर NC से जोड़ देती है।

7.26 रिले के प्रकार (Types of Relay)

Relay सामान्यत: दो प्रकार की होती है—

1. Latching Relay

यह वह Relay है, जिसे हम विद्युत Supply देकर Activate करते हैं। उसके बाद रिले को अगर निष्क्रिय यानी कि विद्युत सप्लाई न दी जाए तो भी उसकी पोजीशन परिवर्तित नहीं होती यानि कि उसकी पोजीशन वहीं पर रुक जाती है, जहाँ से उसे विद्युत देकर एक्टिवेट किया था विद्युत सप्लाई न देने के बाद भी उसकी पोजीशन वापस उसी जगह पर नहीं आती है।

चित्र 7.23 : रिले के प्रकार

2. Non-Latching Relay

यह वह रिले है, जिसे विद्युत देने और न देनें से उनकी Position (स्थिति) बदलती रहती है।

वैसे तो रिले के प्रकार बहुत मात्रा में हैं। इनमें से कुछ प्रकार के बारे में हम जानते हैं—

Electro-magnetic attraction type relays, Rectifier relay, PMMC relay, Gas actual relay, Numerial and micro-processor based relay, Reed switch relay, Static relay, Solid state relay, Frequency monitoring relay, motor load monitoring relay, Liquid monitoring relay, Machine tool relay, Mercury relay, Coaxial relay, Contactor relay, Mercury wetted relay, Multivoltage relay, Safety relay, Polarized relay, Over voltage relay, Time delay relay, Over current relay, Vacuum relay, Buchholz relay and negative resistance relay.

7.27 Relay के उपयोग (Uses of Relay)

1. एक ही नियंत्रण Circuit द्वारा एक या एक से अधिक Circuits (परिपथों) को ON Or OFF करना।
2. किसी Circuit से विद्युतीय रूप से बिना जुड़े हुए भी उसे Control करने में सक्षम होती है।
3. कम पॉवर खर्च करके बहुत अधिक विद्युत शक्ति को Control कर सकते हैं।
4. सभी automatic उपकरणों में रिले का उपयोग किया जाता है, लेकिन कुछ उपकरणों में रिले का उपयोग नहीं किया जाता है, इसका सबसे अच्छा उदाहरण विद्युत प्रेस (Electric iron) है।

प्रश्नावली

1. माइक्रोकंटोलर से सेंसर (Sensor) को किस प्रकार जोड़ेंगे? व्याख्या कीजिए।
2. माइक्रोकंटोलर से 7-सेगमेन्ट डिस्प्ले किस प्रकार जोड़ेंगे? व्याख्या कीजिए।
3. माइक्रोकंटोलर से LED और LCD किस प्रकार जोड़ेंगे?
4. रिले को माइक्रोकंटोलर के साथ इंटरफेस करने की व्याख्या कीजिए।

अध्याय

8

इंटरनेट ऑफ थिंग्स
(Internet of Things)

SYLLABUS

Introduction, internet of things, application, architecture, protocols, functional blocks of IoT, characteristics of IoT, brief idea of Ardunio IDE.

8.1 परिचय (Introduction)

आधुनिकता के इस दौर में तकनीकी और विज्ञान ने असीम तरक्की की हैं। कम्प्युटर, इन्टरनेट, तकनीकी, आधुनिक समाज में शायद ही कोई व्यक्ति होगा, जो इन शब्दों से अनजान होगा या इनके सम्बंध में ना जानता हो। टेक्नोलॉजी तथा विज्ञान ने मानव जीवन को सुगम और सरल बना दिया हैं।

IoT (Internet of Things) इसी का एक बेहतरीन उदाहरण है, इंटरनेट ऑफ थिंग्स ने दैनिक जीवन को अत्यधिक आसान एवं आरामदायी बना दिया है। IoT ने हमारे लिए आधुनिक तकनीकों एवं उपकरणों का उपयोग करके और भी अधिक आसान बना दिया हैं।

इन दिनों हम देखते है, मॉल में हमारे प्रवेश और निकास करते समय दरवाजें स्वत: ही खुलते एवं बंद हो जाते हैं। वातानुकूलित रूम की AC का स्वतः ऑन-ऑफ होना, कार के दरवाजों का स्वतः ओपन होना, ये सब-कुछ ही हमारे लिए किसी आश्चर्य की तरह होता है।

लेकिन क्या हमने कभी सोचा है, कि यह सब सम्भव कैसे है या वह कौन-सी तकनीक है, जिसके माध्यम से यह सम्भव हुआ?

इसी आश्चर्यजनक तकनीक को IoT (इंटरनेट ऑफ थिंग्स) के नाम से जाना जाता है। इंटरनेट ऑफ थिंग्स विज्ञान की वह टेक्नोलॉजी है, जिसके माध्यम से कई टेक्नोलॉजी एवं डिवाइसो को इन्टरनेट के माध्यम से आपस में जोड़ा जाता है। इस टेक्नोलॉजी में इंटरनेट सेंसर द्वारा डिवाइस को आपस में जोड़ा जाता हैं।

8.2 इंटरनेट ऑफ थिंग्स (IoT) क्या है? (What is Internet of Things?)

इंटरनेट ऑफ थिंग्स (IoT) उन उपकरणों का एक समूह है, जो इंटरनेट से जुड़े होते हैं। IoT उपकरणों में Wireless senser, software, actuators और computer devices शामिल होते हैं। वे एक विशेष वस्तु से जुड़े होते हैं जो इंटरनेट के माध्यम से डाटा को स्थानान्तरित करते है।

आज तकनीकी के नए दौर में, IoT devices का उपयोग प्रति दिन हमारे दैनिक जीवन मे किया जा रहा है। जैसे—Smartphone, TV, Lights, AC, Doors, Cars इत्यादि IoT सेंसर द्वारा उपयोग करते हैं। Amazon Echo, Ring DooRell और Nest Thermostat ये सभी आपके इस इंटरनेट ऑफ़ थिंग्स (IoT) का एक हिस्सा हैं। लेकिन यह सिर्फ IoT उपभोक्ता उत्पादों तक सीमित नहीं है बल्कि IoT तकनीक चिकित्सा क्षेत्र से लेकर कृषि उद्योग और दूसरे बड़े क्षेत्रों में उपयोग हो रहा है। Gartner के अनुसार सन् 2021 तक, इंटरनेट ऑफ थिंग्स को बनाने में 25 बिलियन तक उपकरण का निर्माण हो सकते हैं, और सन् 2023 तक, इसकी संख्या तीन गुना तक हो सकती हैं।

उदाहरण—आपकी कार में IoT सिस्टम द्वारा आगे आने वाले रास्ते मे ट्रैफ़िक की पहचान करता है और स्वचालित रूप से व्यक्ति को संदेश भेजता है, जिससे आप अपना समय बचा सकते हैं। स्मार्ट माइक्रोवेव से, जो स्वचालित रूप से सही समय पर भोजन को पकाती हैं, सेल्फ-ड्राइविंग कार, जिनके जटिल सेंसर द्वारा उनके रास्ते में objects का पता लगाते हैं, पहनने योग्य फिटनेस Devices जो आपकी हृदय गति को मापते हैं और उस दिन चले गए कदमों की संख्या, फिर इन सबकी जानकारी का उपयोग करके एक Exercise plans के बारे मे सुझाव भी देता है। यह सब Sensor की मदद से संभव होता है।

8.3 IoT का इतिहास (History of IoT)

स्मार्ट डिवाईसों को आपस में जोड़कर स्मार्ट नेटवर्क बनाने की शुरुआत Carnegie Mellon University में सन 1982 से हो गई थी। इस विश्वविद्यालय में एक Coke Vending Machine बनाई गई थी, जो मशीन के अंदर रखी बोतलों के बारे में बता सकती थी, यह ठण्डी है या नहीं और साथ ही स्टॉक की जानकारी देने में भी सक्षम थी।

लेकिन, Internet of Things टर्म के जनक माननीय केविन एश्टॉन (Kevin Ashton) माने जाते है, जो प्रॉक्टर & गेम्बलर में काम करते थे (बाद में MIT's Auto-ID Center में काम किया)। उन्होने सन् 1999 में Internet for Things शब्द उपयोग किया था। इसी शब्द से **इंटरनेट ऑफ थिंग्स** बना। इस कार्य के लिए उन्होने RFID – Radio-Frequency Identification तकनीक की सिफारिश की थी, जो कम्प्युटरों को अलग-अलग डिवाइसों को प्रबंध करने योग्य क्षमता प्रदान कर सकती थी।

इसके बाद आज तक यह स्मार्ट नेटवर्क तकनीक विकास के दौर से गुजर रही है और भविष्य में इसका विस्तार बहुत व्यापक स्तर पर होने वाला है क्योंकि इंटरनेट ऑफ थिंग्स एक भविष्य की तकनीक है। जो मानव को विज्ञान फंतासी की फिल्मी दुनिया को वास्तविकता बनाने की क्षमता समाए हुए है।

8.4 IoT कैसे काम करता है? (How IoT Works?)

इंटरनेट ऑफ थिंग्स को आधुनिक टेक्नोलॉजी द्वारा एडवांस बनाया जाता हैं। इंटरनेट ऑफ थिंग्स को उपयोग के लिए computer और Internet की सामान्य जानकारी होना आवश्यक है, क्योंकि यह एक नेटवर्किंग टेक्नोलॉजी है। कोई भी व्यक्ति इंटरनेट ऑफ थिंग्स (IOT) की सहायता से अपने घर की डिवाइसो को इंटरनेट की मदद से एक साथ कनेक्ट कर सकता है, जिससे आप उन सब डिवाइसो को कही से भी नियंत्रित कर सकते हैं। इसका उपयोग तब भी कर सकते हैं, जब आपके Mobile और Devices के IP Address एक साथ उपकरण जुड़े हुए हों।

इसका उपयोग करने के लिए उपकरणों को WiFi या Bluetooth के माध्यम से आपस में कनेक्ट किया जाता है। अब इन उपकरणों के बीच वायरलेस तकनीकी माध्यम से डाटा अथवा निर्देशों का आदान-प्रदान होता है। मशीन तकनीक अथवा कम्प्यूटर प्रोग्रामिंग की मदद से इस तकनीकी को पूरी तरह ऑटोमेटिक भी किया जा सकता है, इसके लिए इसमे सेंसर का उपयोग किया जाता है।

उदाहरण—आप अपने घर में कम्प्यूटर पर काम कर रहे हैं और किसी आपातकालीन परिस्थिती के कारण आपको बाहर जाना पड़ा, और इस जल्दबाजी में आप अपना कम्प्यूटर बंद करना भूल जाते हैं। अपने घर से निकलने के कुछ समय बाद अचानक आपको याद आता है, कि आप अपना कम्प्यूटर बंद करना भूल गये हैं। इस परिस्थिती में इंटरनेट ऑफ थिंग्स आपके लिए एक उपयोगी तकनीक साबित हो सकती हैं। इंटरनेट ऑफ थिंग्स के माध्यम से आप जहाँ कहीं मौजूद हो वहाँ से अपने कम्प्यूटर को शटडाउन कर सकते हैं।

इंटरनेट ऑफ थिंग्स तकनीक का आपातकालीन परिस्थितीयों में विशेष योगदान है। यदि किसी रोगी को हॉस्पिटल में लाईफ सपोर्टिंग सिस्टम पर रखा जाता है, और यदि अचानक रोगी का स्थिती बिगडने लगे तब इन परिस्थितीयों में रोगी के महत्वपूर्ण लक्षणों को मॉनिटर करके किसी भी आपातकाल की सूचना तुरन्त डॉक्टर तक पहुँचा देता हैं।

8.5 IoT के अनुप्रयोग (Applications of IoT)

8.5.1 स्मार्ट होम में IoT (IoT in Smart Home)

स्मार्ट होम IoT Application की सहायता से हम घरेलु उपकरणों को स्मार्ट फोन अथवा सेंसर की सहायता से कनेक्ट कर सकते हैं। इसकी सहायता से हम घर में आने से पहले घर की Lights और AC चालू कर सकते है, और घर से निकलने के बाद उन्हें बंद कर सकते हैं। यदि आप अपने घर का दरवाजा बंद करना भूल गए हैं और उस दरवाजे में Sensor लगा है, तो थोड़े समय बाद सेंसर की मदद से दरवाजा अपने आप बंद हो जाएगा और इसका मैसेज आपके स्मार्टफोन पर भी आ जाता हैं।

8.5.2 पहने जाने वाले डिवाइस में IoT (IoT Wearable Devices)

Wearable तकनीक उन उपकरणों, जिन्हे आप पहन सकते हैं, जैसे—स्मार्ट क्लॉथ, स्मार्ट वॉच, स्मार्ट शूज आदि, इंटरनेट ऑफ थिंग्स के अंतर्गत आती है। यह फिटनेस, स्वास्थ्य और मनोरंजन के लिए उपयोगी होती हैं। आप Smart watch के बारे मे तो जानते ही होंगे।

उदाहरण—Samsung, Apple या Google कम्पनी की Smart watch, जिसमें बहुत-से फीचर्स होते हैं ऐसी घड़ी मे सेंसर लगे होते हैं, जिसको आप अपने मोबाइल से कनेक्ट कर सकते हैं और जिम में वर्कआउट करते समय ईमेल, कॉल, सांग्स आदि सुन सकते है और साथ ही यह Smart Watch आपको आपकी सेहत के बारे में भी बताती हैं, जिससे आपको मोबाइल को बार-बार लेने की जरूरत नहीं पड़ती हैं।

8.5.3 औद्योगिक इंटरनेट IoT (IoT Industrial Internet)

औद्योगिक सेक्टर मे इंडस्ट्रियल इंटरनेट की बड़ी चर्चा है, जिसे इंडस्ट्रियल इंटरनेट ऑफ थिंग्स (IIoT) कहते हैं। इंडस्ट्रियल इंटरनेट ऑफ थिंग्स (IIoT) का प्रयोग मशीनों को बनाने के लिए Sensors, Software, Devices और बड़े Data analytics के साथ डाटा कलेक्शन, एक्सचेंज एवं एनालाईज में बहुत उपयोगी है और यह इंडस्ट्रियल इंजीनियरिंग को मजबूत बनाने मे बड़ा योगदान दे रहा है। स्मार्ट मशीनें डाटा के माध्यम से संचार करने में मनुष्यों की तुलना में अधिक Accurate और Consistent परिणाम देती हैं।

8.5.4 कृषि में IoT (IoT in Agriculture)

इन्टरनेट ऑफ थिंग्स की सहायता से मौसम का अनुमान लगाया जा सकता हैं। IoT की सहायता से खाद्य-फल की उपलब्धता और आवश्यकता का रिकार्ड रखा जा सकता हैं। किसानों को अच्छी फसलों का उत्पादन करने के लिए मिट्टी की अच्छी गुणवत्ता महत्वपूर्ण होती है इसलिए इंटरनेट ऑफ थिंग्स (IoT) किसानों को अपने मिट्टी की स्थिति की महत्वपूर्ण जानकारी पाने मे बहुत मदद मिलती है।

जैसे—मिट्टी में नमी, अम्लता का स्तर, कुछ पोषक तत्वों की उपस्थिति, तापमान और कई अन्य रासायनिक विशेषताओं का जानकारी, किसानों को सिंचाई नियंत्रित जानकारी, बुवाई शुरू करने के लिए सबसे अच्छा समय बताती है और यहाँ तक कि पौधों और मिट्टी की बीमारियों की उपस्थिति का भी पता लगाती है।

8.5.5 कार में IoT (IoT in Connected Car)

Connected Car वह वाहन है, जो स्वयं संचालन, रखरखाव के साथ-साथ ऑनबोर्ड सेंसर और इंटरनेट कनेक्टिविटी का उपयोग करके यात्रियों के आराम के लिए अनुकूल होता हैं।

इन्टरनेट ऑफ थिंग्स का यह एप्लिकेशन कार के भीतर की फंक्शन को संचालित करने में सहायक होता है। इसकी सहायता से कार की गति को नियंत्रित किया जा सकता है, दरवाजों को खोलने एवं बंद करने और लाईटों को ऑन-ऑफ आदि कर सकते हैं।

अधिकांश बड़ी ऑटोमोबाइल कम्पनियाँ कनेक्टेड कार के समाधान पर कार्य कर रही हैं, जैसे—टेस्ला, बीएमडब्ल्यू, एप्पल, गूगल प्रमुख ब्रांड ऑटोमोबाइल में अगली नयी फ़ीचर्स लाने पर काम कर रहे हैं।

8.5.6 स्मार्ट सिटी में IoT (IoT in Smart City)

स्मार्ट सिटी एक शक्तिशाली एप्लीकेशन है। इंटरनेट ऑफ थिंग्स की सहायता से दैनिक परेशानियों, जैसे—जल, बिजली आपूर्ती, ट्रैफीक, क्राइम एवं पर्यावरण सम्बन्धित समस्याओं का निवारण किया जा सकता हैं। Internet Of Things की मदद से Web Application मे Sensor को इंस्टॉल करके उपलब्ध फ्री-पार्किंग के Slots पा सकते हैं।

8.5.7 अस्पताल में IoT (IoT in Hospitals)

Hospitals और स्वास्थ्य क्षेत्र में इंटरनेट ऑफ थिंग्स की महत्ता अतुलनीय हैं। अस्पताल के उपकरण और डाटा का ब्यौरा रखने के साथ-साथ मरीज के रोग के लक्षण और रिपोर्टस् का रिकार्ड रखा जाता हैं। IoT मरीज की स्थिती को मोनिटर करके किसी भी आपातकालीन सुचना को तुरंत डॉक्टर तक पहुँचाने में भी सहायक सिद्द होता हैं।

8.6 IoT की विशेषताए (Features of IoT)

कनेक्टिविटी (Connectivity)—कनेक्टिविटी का अर्थ है, IoT प्लेटफॉर्म से IoT का सभी उपकरणों के बीच एक अच्छा कनेक्शन स्थापित करना जो सर्वर या क्लाउड हो सकता है। IoT उपकरणों को जोड़ने के बाद विश्वसनीय, सुरक्षित और कम्युनिकेशन के लिए उपकरणों तथा क्लाउड के बीच हाई स्पीड डाटा संचार की आवश्यकता होती हैं।

ऐनालाईजिंग (Analyzing)—IoT उपकरणों को कनेक्ट करने के बाद, डिवाइस में एकत्रित हुए डाटा का वास्तविक समय में विश्लेषण करके उसे Business intelligence निर्माण के लिए उपयोग करते हैं।

आर्टिफीशियल इन्टेलिजेस (Artificial Intelligence)—IoT टेक्नोलॉजी Hardware, Calculation और Software एक साथ आता है, जो IoT डिवाइस को स्मार्ट बनाता है। यह वातावरण में अपनी क्षमताओं को बढ़ाता है, जो उपकरणों को किसी विशेष स्थिति में बुद्धिमान तरीके से आउटपुट देने में आसानी होती है।

सेंसिंग (Sensing)—IoT टेक्नोलॉजी में उपयोग किए जाने वाले Sensor device मौसम में होने वाले परिवर्तन का पता लगाते हैं और उसकी स्थिति के अनुसार डाटा की रिपोर्ट करते हैं। IoT technology नेटवर्क को सक्रिय नेटवर्क बनाता है। Sensor के बिना IoT वातावरण की स्थिति का इफेक्टिव या सही पता नहीं लगा सकता हैं।

सूरक्षा (Security)—IoT devices सुरक्षा की नजर से स्वाभाविक रूप से असुरक्षित हैं, क्योंकि कनेक्टिविटी डिवाइसो के माध्यम से संवेदनशील जानकारी को एंडपॉइंट से दूसरे डिवाइस तक इंटरनेट द्वारा पहुँचाया जाता है। इसलिए इससे सुरक्षा से जुड़ी समस्याओं को भूलना एक प्रकार की गलती होगी। IoT System को बनाते समय बेहतर सुरक्षा, सुरक्षा सम्बन्धी उपायों और फायरवॉल का उपयोग करना होता है, ताकि Data को दुरुपयोग और हेरफेर होने से दूर रखा जा सके।

8.7 IoT के घटक (Components of IoT)

IoT चार मुख्य मुलभूत घटकों पर कार्य करता है—

1. सेंसर (Sensors)—सेंसर अपने आस-पास के वातावरण से Data एकत्रित करता है और इस एकत्रित Data को साधारण तापमान की निगरानी से लेकर Complex वीडियो भी प्रदान करता है। एक Device में कई सेंसर लगे हो सकते हैं। सेंसर उदाहरण जैसे—GPS, Camera, Microphone, Temperature सेंसर आदि होते हैं।

2. कनेक्टिविटी (Connectivity)—Connectivity ट्रांसपोर्ट माध्यम का उपयोग करके एकत्रित Data को Cloud पर भेजा जाता है। इसलिए सेंसर संचार के विभिन्न माध्यमों; जैसे—WiFi, Setellite Network, Bluetooth, Wide Area Network आदि द्वारा क्लाउड से कनेक्टेड होते हैं।

3. डाटा प्रोसेसिंग (Data Processing)—Data processing में एकत्रित हुआ डाटा Cloud पर जाता है, सॉफ्टवेयर अधिग्रहण किये डाटा को प्रोसेस करता है। यह बहुत आसान वस्तुओं से लेकर कुछ भी हो सकता है, जैसे उपकरणों पर Temperature reading की जाँच करना या बहुत कठिन वस्तुओं की पहचान करना इत्यादि।

4. **यूजर इंटरफ़ेस** (User Interface)—डिवाइस की यूज़र्स इंटरफ़ेस पर जानकारी को End user के लिए एकत्रित की गई होती है। इसको alarm, text, email, message के द्वारा सूचित किया जा सकता है। यदि दो डिवाइस आपस में कनेक्टेड हैं, तो Actuators का उपयोग किया जाता है।

8.8 IoT के उदाहरण (Internet of Things Examples)

1. **स्मार्ट लॉक** (Smart Lock)—स्मार्ट लॉक को स्मार्ट वॉच या स्मार्ट फोन से कनेक्ट कर लेते हैं, और इन्ही उपकरणों की सहायता से इन स्मार्ट लॉक को लॉक और अनलॉक किया जाता हैं। इन्हे लॉक और अनलॉक करने के लिए अतिरिक्त चाभी की जरूरत नही होती है।

2. **आटोमेटिक कार ट्रैकिंग अडॉप्टर** (Automatic Car Tracking Adapter)—Automatic Car Tracking Adapter कार से संबंधित जानकारियों को ट्रैक करता हैं। यह डिवाइस वाहन की रनिंग हावर्स, ईंधन खपत, माईलेज, इग्निषन ऑन-ऑफ संबंधित जानकारियों को ट्रैक करके वाहन के स्वामी को सूचित करता हैं।

3. **स्मार्ट थर्मोस्टेट** (Smart Thermostat)—स्मार्ट थर्मोस्टेट घर की हीटिंग और एयर कंडिशनिंग के लिए उपयोगी हैं। दिन-भर के लिए सेड्यूल करके घर के तापमान को नियंत्रित कर सकते हैं, इसके अतिरिक्त इसे सेंसर या किसी एडिशनल डिवाइस; जैसे—स्मार्ट वॉच या स्मार्ट फोन की मदद से भी प्रयोग किया जा सकता हैं।

4. **एक्टिविटी ट्रैकर** (Activity Tracker)—एक्टिविटी ट्रैकर को फिटनेस ट्रैकर नाम से भी जाना जाता है। इस IoT एप्लिकेशन को स्मार्ट वॉच या स्मार्ट फोन में इंस्टाल किया जाता है। यह हमारी दिन-भर की गतिविधियों को मॉनिटर करता हैं; जैसे—चलना, दौड़ना तथा इसके आधार पर स्वास्थ्य सम्बंधित अवलोकन करता है, जैसे—पल्स रेट, कैलोरी एवं कोलेस्ट्रोल आदि।

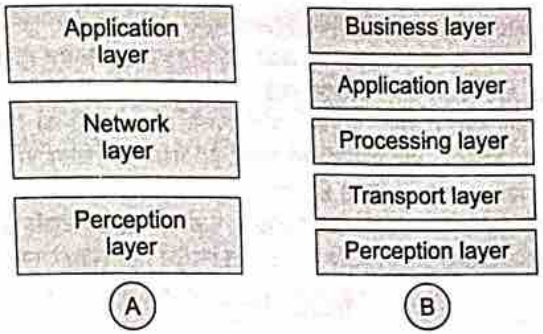
8.9 IoT का आर्किटेकचर (Architecture of IoT)

IoT के लिए आर्किटेकचर पर एक भी आम सहमति नहीं है, जो सार्वभौमिक रूप से सहमत है। विभिन्न शोधकर्ताओं द्वारा विभिन्न आर्किटेकचर का प्रस्ताव किया गया है।

8.9.1 तीन और पांच लेयेर आर्किटेकचर (Three- and Five-Layer Architectures)

सबसे बुनियादी आर्किटेकचर तीन-परत आर्किटेकचर है। जैसा कि चित्र में दिखाया गया है। यह इस क्षेत्र में अनुसंधान के शुरुआती चरणों में पेश किया गया था। इसकी तीन परतें हैं, अर्थात्, धारणा, नेटवर्क, और अनुप्रयोग परतें।

1. धारणा परत (Perception Layer) एक भौतिक परत है, जिसमें संवेदन और पर्यावरण के बारे में जानकारी एकत्र करने के लिए सेंसर हैं। यह कुछ भौतिक मापदंडों को महसूस करता है या वातावरण में अन्य स्मार्ट वस्तुओं की पहचान करता है।
2. नेटवर्क परत (The Network Layer) अन्य स्मार्ट वस्तुओं, नेटवर्क उपकरणों और सर्वर से कनेक्ट करने के लिए जिम्मेदार है। इसकी विशेषताओं का उपयोग सेंसर डाटा को प्रसारित करने और संसाधित करने के लिए भी किया जाता है।
3. उपयोगकर्ता को एप्लिकेशन विशिष्ट सेवाएँ प्रदान करने के लिए एप्लिकेशन परत (Application layer) जिम्मेदार है। यह विभिन्न अनुप्रयोगों को परिभाषित करता है, जिसमें इंटरनेट ऑफ थिंग्स को तैनात किया जा सकता है, उदाहरण के लिए, स्मार्ट होम, स्मार्ट सिटी और स्मार्ट हेल्थ।



चित्र 8.1 : Architecture of IoT (A : three layers) (B : five layers)

थ्री-लेयर आर्किटेक्चर IoT के मुख्य विचार को परिभाषित करता है, लेकिन यह IoT पर शोध के लिए पर्याप्त नहीं है, क्योंकि शोध अक्सर IoT के बारीक पहलुओं पर केंद्रित होता है। यही कारण है, कि हमारे पास साहित्य में प्रस्तावित कई और अधिक स्तरित आर्किटेक्चर हैं। एक पाँच-परत आर्किटेक्चर है, जिसमें अतिरिक्त रूप से प्रसंस्करण और व्यावसायिक परतें शामिल हैं। पाँच परतीय धारणा, परिवहन, प्रसंस्करण, अनुप्रयोग और व्यावसायिक परतें हैं (चित्र 8.1B को देखें)। धारणा और अनुप्रयोग परतों की भूमिका तीन परतीय वास्तुकला के समान है। हम शेष तीन परतों के कार्य को रेखांकित करते हैं।

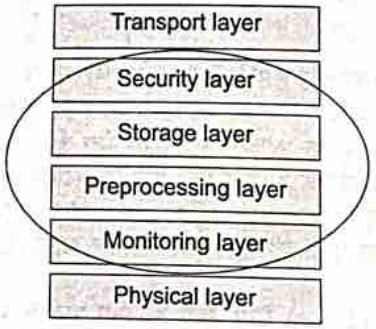
1. **परिवहन परत** (Transport Layer) सेंसर डाटा को बोध परत से प्रसंस्करण परत और वायरलेस, 3 जी, लैन, ब्लूटूथ, RFID और NFC जैसे नेटवर्क के माध्यम से स्थानांतरित करती है।
2. **प्रोसेसिंग लेयर** (Processing Layer) को मध्य परत के नाम से भी जाना जाता है। यह ट्रांसपोर्ट लेयर से आने वाले भारी मात्रा में डाटा को स्टोर, एनालिसिस और प्रोसेस करता है। यह निचली परतों में सेवाओं के विविध सेट का प्रबंधन और प्रदान कर सकता है। यह डाटाबेस, क्लाउड कम्प्यूटिंग और बिग डाटा प्रोसेसिंग मॉड्यूल जैसी कई तकनीकों को नियोजित करता है।
3. **व्यावसायिक परत** (Business Layer) सम्पूर्ण IoT प्रणाली का प्रबंधन करती है, जिसमें अनुप्रयोग, व्यवसाय और लाभ मॉडल, और उपयोगकर्ताओं की गोपनीयता शामिल है। व्यावसायिक परत इस पत्र के दायरे से बाहर है। इसलिए, हम इसके बारे में आगे चर्चा नहीं करते हैं।

निंग और वांग द्वारा प्रस्तावित एक और आर्किटेक्चर मानव मस्तिष्क में प्रसंस्करण की परतों से प्रेरित है। यह मानव की बुद्धि और सोचने, महसूस करने, याद रखने, निर्णय लेने और भौतिक वातावरण पर प्रतिक्रिया करने की क्षमता से प्रेरित है। इसका गठन तीन भागों में किया जाता है। पहला मानव मस्तिष्क है, जो प्रसंस्करण और डाटा प्रबंधन इकाई या डाटा सेंटर के अनुरूप है। दूसरा रीढ़ की हड्डी है, जो डाटा प्रोसेसिंग नोड्स और स्मार्ट गेटवे के वितरित नेटवर्क के अनुरूप है। तीसरा तंत्रिकाओं का नेटवर्क है, जो नेटवर्किंग घटकों और सेंसर से मेल खाती है

8.9.2 क्लाउड और फोग पर आधारित आर्किटेक्चर (Cloud and Fog Based Architectures)

आइए, अब दो प्रकार के सिस्टम आर्किटेक्चर पर चर्चा करते हैं: क्लाउड और फॉग कंप्यूटिंग, ध्यान दें कि यह वर्गीकरण पिछले सेक्शन के वर्गीकरण से अलग है। जो कि प्रोटोकॉल के आधार पर किया गया था। विशेष रूप से, हम IoT उपकरणों द्वारा प्राप्त डाटा की प्रकृति, और डाटा प्रोसेसिंग की प्रकृति के बारे में थोड़ा अस्पष्ट रहे हैं। कुछ सिस्टम आर्किटेक्चर में क्लाउड कम्प्यूटर द्वारा डाटा प्रसंस्करण बड़े केंद्रीकृत फैशन में किया जाता है। इस तरह के क्लाउड सेंट्रिक आर्किटेक्चर क्लाउड को केंद्र में रखता है, इसके ऊपर के अनुप्रयोग और इसके नीचे स्मार्ट चीजों का नेटवर्क क्लाउड कंप्यूटिंग को प्रधानता दी जाती है, क्योंकि यह महान लचीलापन और मापनीयता प्रदान करता है। यह कोर इंफ्रास्ट्रक्चर, प्लेटफॉर्म, सॉफ्टवेयर और स्टोरेज जैसी सेवाएँ प्रदान करता है। डेवलपर्स अपने स्टोरेज टूल, सॉफ्टवेयर टूल, डाटा माइनिंग और मशीन लर्निंग टूल्स और क्लाउड के माध्यम से विजुअलाइज़ेशन टूल प्रदान कर सकते हैं।

हाल ही में, एक अन्य सिस्टम आर्किटेक्चर, अर्थात् फोग कंप्यूटिंग की ओर एक कदम है, जहाँ सेंसर और नेटवर्क गेटवे डाटा प्रोसेसिंग और एनालिटिक्स का एक भाग है। फॉग आर्किटेक्चर एक स्तरित दृष्टिकोण प्रस्तुत करता है जैसा कि चित्र में दिखाया गया है, जो भौतिक और परिवहन परतों के बीच निगरानी, प्रीप्रोसेसिंग, भंडारण और सुरक्षा परतों को सम्मिलित करता है। निगरानी परत बिजली, संसाधनों, प्रतिक्रियाओं और सेवाओं की निगरानी करती है। प्रीप्रोसेसिंग परत सेंसर डाटा के फ़िल्टरिंग, प्रोसेसिंग और एनालिटिक्स करती है। अस्थायी भंडारण परत डाटा प्रतिकृति की वितरण और भंडारण जैसी भंडारण कार्यात्मकता प्रदान करती है। अंत में, सुरक्षा परत एन्क्रिप्शन / डिक्रिप्शन करता है और डाटा अखंडता और गोपनीयता सुनिश्चित करता है। क्लाउड पर डाटा भेजने से पहले नेटवर्क के किनारे पर निगरानी और प्रीप्रोसेसिंग की जाती है।



चित्र 8.2 : Fog Architecture of a Smart IoT Gateway

अक्सर शब्द 'फॉग कंप्यूटिंग' और 'एज कंप्यूटिंग' का उपयोग एक-दूसरे से किया जाता है। बाद वाला शब्द पूर्व को दर्शाता है और अधिक सामान्य माना जाता है। सिस्को द्वारा मूलतः फॉग कंप्यूटिंग को स्मार्ट गेटवे और स्मार्ट सेंसर कहा जाता है, जबकि एज कंप्यूटिंग प्रकृति में थोड़ा अधिक मर्मज्ञ है। यह प्रतिमान मोटर, पंप, या रोशनी जैसे भौतिक उपकरणों में स्मार्ट डाटा प्रीप्रोसेसिंग क्षमताओं को जोड़ता है। इसका उद्देश्य इन उपकरणों में डाटा का अधिक-से-अधिक प्रीप्रोसेसिंग करना है, जिन्हें नेटवर्क के किनारे पर होना कहा जाता है। सिस्टम आर्किटेक्चर के संदर्भ में, आर्किटेक्चर आरेख चित्र से प्रशंसनीय रूप से भिन्न नहीं है। परिणामस्वरूप, हम अलग से एज कंप्यूटिंग का वर्णन नहीं करते हैं।

अंत में, प्रोटोकॉल आर्किटेक्चर और सिस्टम आर्किटेक्चर के बीच का अंतर बहुत कुरकुरा नहीं है। अक्सर प्रोटोकॉल और सिस्टम कोड किए जाते हैं। हम फॉग और क्लाउड आर्किटेक्चर दोनों के लिए जेनरिक 5-लेयर IoT प्रोटोकॉल स्टैक का उपयोग करेंगे।

8.10 IoT प्रोटोकॉल (IoT Protocol)

एक प्रोटोकॉल नियमों का एक मानक सेट होता है जो इलेक्ट्रॉनिक उपकरणों को एक-दूसरे के साथ संवाद करने की अनुमति देता है। इन नियमों में शामिल होता है कि किस प्रकार के डेटा को प्रेषित किया जा सकता है, डेटा भेजने और प्राप्त करने के लिए कौन-सी कमांड का उपयोग किया जाता है और डेटा ट्रांसफर की पुष्टि कैसे की जाती है। IoT प्रोटोकॉल IoT प्रौद्योगिकी स्टैक का एक महत्वपूर्ण हिस्सा है, उनके बिना हार्डवेयर बेकार हो जाएगा क्योंकि IoT प्रोटोकॉल इसे संरचित और सार्थक तरीके से डेटा का आदान-प्रदान करने में सक्षम बनाते हैं। डेटा के इन स्थानान्तरित टुकड़ों में से, अंतिम उपयोगकर्त्ता के लिए उपयोगी जानकारी निकाली जा सकती है। इंटरनेट ऑफ थिंग्स के बारे में बात करते समय, हम हमेशा संचार के बारे में सोचते हैं। सेंसर, डिवाइस, गेटवे, सर्वर और उपयोगकर्त्ता एप्लिकेशन्स के बीच इंटरैक्शन आवश्यक विशेषता है जो इंटरनेट ऑफ थिंग्स को बनाता है। लेकिन क्या बात करने और बातचीत करने के लिए इस सभी स्मार्ट कम्पोनेंट को सक्षम बनाता है IoT प्रोटोकॉल जो भाषाओं के रूप में देखा जा सकता है जो संचार करने के लिए IoT प्रोटोकॉल का उपयोग करता है।

8.10 IoT प्रोटोकॉल के प्रकार (Types of IoT Protocols)

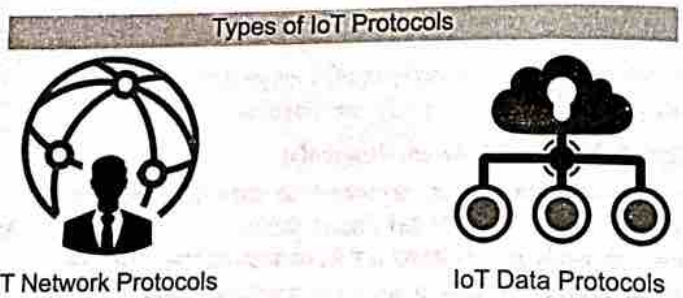
IoT प्रोटोकॉल और मानकों को दो अलग-अलग श्रेणियों में वर्गीकृत किया जा सकता है।

1. IoT नेटवर्क प्रोटोकॉल **2. IoT डाटा प्रोटोकॉल**

8.10.1 IoT नेटवर्क प्रोटोकॉल (IoT Network Protocols)

IoT नेटवर्क प्रोटोकॉल का उपयोग नेटवर्क पर उपकरणों को जोड़ने के लिए किया जाता है। इंटरनेट पर उपयोग किए जाने वाले संचार प्रोटोकॉल का सेट हैं। IoT नेटवर्क प्रोटोकॉल का उपयोग करते हुए, नेटवर्क के दायरे में एंड-टू-एंड डाटा संचार की अनुमति होती है। विभिन्न IoT नेटवर्क प्रोटोकॉल निम्नलिखित हैं—

1. **HTTP** (HyperText Transfer Protocol)—हाइपरटेक्स्ट ट्रांसफर प्रोटोकॉल IoT नेटवर्क प्रोटोकॉल का सबसे अच्छा उदाहरण है। इस प्रोटोकॉल ने वेब पर डाटा संचार की नींव बनाई है। यह सबसे मुख्य प्रोटोकॉल है, जिसका उपयोग IoT उपकरणों के लिए किया जाता है। जब बहुत अधिक डाटा प्रकाशित किया जाना है। हालांकि, HTTP प्रोटोकॉल को इसकी लागत, बैटरी-जीवन, ऊर्जा की बचत और अधिक बाधाओं के कारण पसंद नहीं किया जाता है। Additive विनिर्माण / 3D प्रिंटिंग HTTP प्रोटोकॉल के उपयोग मामलों में से एक है। यह कम्प्यूटर को नेटवर्क में 3D प्रिंटर कनेक्ट करने और 3-D वस्तुओं और पूर्व-निर्धारित प्रक्रिया प्रोटोटाइप को प्रिंट करने में सक्षम बनाता है।
2. **LoRaWAN (Long Range Wide Area Network)**—यह एक लम्बी दूरी, कम पॉवर का प्रोटोकॉल है, जो शोर स्तर के नीचे सिग्नल का डिटेक्शन प्रदान करता है। लोरावन बैटरी संचालित वस्तुओं को निजी या वैश्विक नेटवर्क में इंटरनेट से वायरलेस रूप से जोड़ता है। यह संचार प्रोटोकॉल मुख्य रूप से स्मार्ट शहरों में उपयोग किया जाता है, जहाँ लाखों डिवाइस होते हैं, जो कम शक्ति और मेमोरी के साथ कार्य करती है। स्मार्ट स्ट्रीट लाइटिंग लोरावन आईओटी प्रोटोकॉल का व्यावहारिक उपयोग है। इस प्रोटोकॉल का उपयोग करके स्ट्रीट लाइट को लोरा गेटवे से जोड़ा जा सकता है। प्रवेश द्वार, बदले में, क्लाउड एप्लिकेशन से कनेक्ट होता है, जो परिवेश प्रकाश के आधार पर प्रकाश बल्ब की तीव्रता को स्वचालित रूप से नियंत्रित करता है, जो दिन के समय बिजली की खपत को कम करने में मदद करता है।
3. **ब्लूटूथ (Bluetooth)**—कम दूरी के संचार के लिए ब्लूटूथ व्यापक रूप से उपयोग किए जाने वाले प्रोटोकॉल में से एक है। यह वायरलेस डाटा ट्रांसमिशन के लिए एक मानक IoT प्रोटोकॉल है। यह संचार प्रोटोकॉल इलैक्ट्रॉनिक उपकरणों के बीच कम दूरी, कम शक्ति, कम लागत और वायरलेस ट्रांसमिशन के लिए सुरक्षित और परिपूर्ण है। BLE (ब्लूटूथ लो एनर्जी) ब्लूटूथ प्रोटोकॉल का एक कम-ऊर्जा का संस्करण है, जो बिजली की खपत को कम करता है और IoT उपकरणों को जोड़ने में महत्वपूर्ण भूमिका निभाता है। ब्लूटूथ प्रोटोकॉल का उपयोग अधिकतर स्मार्ट वियरेबल, स्मार्टफोन और अन्य मोबाइल उपकरणों में किया जाता है, जहाँ उच्च शक्ति और मेमोरी के बिना डाटा के छोटे टुकड़ों का आदान-प्रदान किया जा सकता है। उपयोग में आसान ब्लूटूथ IoT डिवाइस कनेक्टिविटी प्रोटोकॉल की सूची में सबसे ऊपर है।
4. **जिगबी (ZigBee)**—ZigBee एक ऐसा IoT प्रोटोकॉल है जो स्मार्ट ऑब्जेक्ट्स को एक साथ कार्य करने की अनुमति देता है। यह होम ऑटोमेशन में उपयोग किया जाता है। औद्योगिक सेटिंग्स के लिए अधिक प्रसिद्ध, ZigBee का उपयोग उन ऐप्स के साथ किया जाता है, जो कम दूरी के बीच कम-दर डाटा हस्तांतरण का समर्थन करते हैं। शहरी क्षेत्रों में स्ट्रीट लाइटिंग और बिजली के मीटर, जो कम बिजली की खपत प्रदान करते हैं, उनमें ZigBee संचार प्रोटोकॉल का उपयोग किया जाता है। इसका उपयोग सुरक्षा प्रणालियों और स्मार्ट घरों में भी किया जाता है।



चित्र 8.3

8.10.2 IoT डाटा प्रोटोकॉल (IoT Data Protocols)

IoT डाटा प्रोटोकॉल कम बिजली IoT उपकरणों को जोड़ने के लिए उपयोग किया जाता है। यह प्रोटोकॉल किसी भी इंटरनेट कनेक्शन के बिना उपयोगकर्ता की ओर से हार्डवेयर के साथ बिंदु-से-बिंदु संचार प्रदान करता है। IoT डाटा प्रोटोकॉल में कनेक्टिविटी वायर्ड या सेलुलर नेटवर्क के माध्यम से होती है। IoT डाटा प्रोटोकॉल में से कुछ हैं निम्नलिखित—

1. **Message Queue Telemetry Transport (MQTT)**—IoT उपकरणों के लिए सबसे पसंदीदा प्रोटोकॉल में से एक है, MQTT विभिन्न इलैक्ट्रॉनिक उपकरणों से डाटा को एकत्र करता है और दूरस्थ डिवाइस निगरानी का समर्थन करता है। यह एक सदस्यता / प्रकाशन प्रोटोकॉल (Subscribe/publish protocol) है, जो ट्रांसमिशन कंट्रोल प्रोटोकॉल (TCP) पर चलता है, जिसका अर्थ है, कि यह वायरलैस नेटवर्क के माध्यम से इवेंट-संचालित संदेश विनिमय का समर्थन करता है। MQTT मुख्य रूप से उन उपकरणों में उपयोग किया जाता है, जो किफायती हैं और उन्हें कम शक्ति और मेमोरी की आवश्यकता होती है। उदाहरण के लिए, फायर डिटेक्टर, कार सेंसर, स्मार्ट वॉच और टेक्स्ट-आधारित मैसेजिंग के लिए ऐप।
2. **Constrained Application Protocol (CoAP)**—CoAP प्रतिबंधित गैजेट्स के लिए एक इंटरनेट-उपयोगिता प्रोटोकॉल है। इस प्रोटोकॉल का उपयोग करके, क्लाइंट सर्वर को एक अनुरोध भेज सकता है और सर्वर क्लाइंट में प्रतिक्रिया को HTTP में वापस भेज सकता है। हल्के वजन के कार्यान्वयन के लिए, यह यूडीपी (यूजर डाटाग्राम प्रोटोकॉल) का उपयोग करता है और अंतरिक्ष उपयोग को कम करता है। प्रोटोकॉल बाइनरी डाटा प्रारूप EXL (Efficient XML Interchanges) का उपयोग करता है। CoAP प्रोटोकॉल का उपयोग मुख्य रूप से स्वचालन, मोबाइल और माइक्रोकंट्रोलर में किया जाता है। प्रोटोकॉल घरेलू ;उपकरणों जैसे—एप्लिकेशन एंडपॉइंट्स के लिए एक अनुरोध भेजता है और एप्लिकेशन में सेवाओं और संसाधनों की प्रतिक्रिया भेजता है।
3. **Advanced Message Queuing Protocol (AMQP)**—AMQP संदेश-उन्मुख मिडलवेयर (message- oriented middleware) वातावरण के लिए एक सॉफ्टवेयर परत प्रोटोकॉल है, जो रूटिंग और कतारबद्धता प्रदान करता है। इसका उपयोग विश्वसनीय पॉइंट-टू-पॉइंट कनेक्शन के लिए किया जाता है और कनेक्टेड डिवाइस और क्लाउड के बीच डाटा के सहज और सुरक्षित विनिमय का समर्थन करता है। AMQP में तीन अलग-अलग घटक होते हैं; जैसे—एक्सचेंज, मैसेज क्यू और बाइंडिंग। ये सभी तीन घटक संदेशों के सुरक्षित और सफल विनिमय और भंडारण को सुनिश्चित करते हैं। यह एक संदेश के दूसरे के साथ संबंध स्थापित करने में भी मदद करता है। AMQP प्रोटोकॉल का उपयोग मुख्य रूप से बैंकिंग उद्योग में किया जाता है। जब भी कोई संदेश किसी सर्वर द्वारा भेजा जाता है, तो प्रोटोकॉल संदेश को तब तक ट्रैक करता है, जब तक कि प्रत्येक संदेश बिना किसी विफलता के इच्छित उपयोगकर्ताओं / गंतव्यों तक पहुंच न जाए।

4. **Machine-to-Machine (M2M) Communication Protocol**—यह एक खुला उद्योग प्रोटोकॉल (Open industry protocol) है, जो IoT उपकरणों के दूरस्थ अनुप्रयोग प्रबंधन प्रदान करने के लिए बनाया गया है। M2M संचार प्रोटोकॉल लागत प्रभावी हैं और सार्वजनिक नेटवर्क का उपयोग करते हैं। यह एक ऐसा वातावरण बनाता है जहाँ दो मशीनें संचार करती हैं और डाटा का आदान-प्रदान करती हैं। यह प्रोटोकॉल मशीनों की स्व-निगरानी का समर्थन करता है और सिस्टम को बदलते परिवेश के अनुसार अनुकूलित करने की अनुमति देता है। M2M संचार प्रोटोकॉल का उपयोग स्मार्ट घरों, स्वचालित वाहन प्रमाणीकरण, वेल्डिंग मशीनों और एटीएम मशीनों के लिए किया जाता है।
5. **Extensible Messaging and Presence Protocol (XMPP)**—XMPP विशिष्ट रूप से डिजाइन किया गया है। यह वास्तविक समय में संदेशों का आदान-प्रदान करने के लिए एक तंत्र का उपयोग करता है। XMPP लचीला है और मूल रूप से परिवर्तनों के साथ एकीकृत कर सकता है। ओपन XML (एक्सटेंसिबल मार्कअप लैंग्वेज) का उपयोग करके विकसित किया गया, XMPP एक उपस्थिति संकेतक के रूप में काम करता है, जो संदेशों को प्रसारित करने या प्राप्त करने वाले सर्वर या उपकरणों की उपलब्धता स्थिति को दर्शाता है। Google टॉक और व्हाट्सएप जैसे त्वरित मैसेजिंग ऐप के अलावा, XMPP का उपयोग ऑनलाइन गेमिंग, समाचार वेबसाइटों और वॉयस ओवर इंटरनेट प्रोटोकॉल (VoIP) में भी किया जाता है।

8.11 Arduino

Arduino एक ओपन-सोर्स इलैक्ट्रॉनिक्स प्लेटफ़ॉर्म है, जो आसान हार्डवेयर और सॉफ़्टवेयर पर आधारित है। Arduino बोर्ड निम्न इनपुट को पढ़ने में सक्षम हैं, जैसे एक सेंसर पर प्रकाश, एक बटन पर एक उंगली, एक मोटर को सक्रिय करना, एक एलईडी चालू करना, कुछ ऑनलाइन प्रकाशित करना। आप अपने बोर्ड को बता सकते हैं, कि बोर्ड पर माइक्रोकंट्रोलर को निर्देशों का एक सेट भेजकर क्या करना है। ऐसा करने के लिए आप संसाधन के आधार पर Arduino प्रोग्रामिंग भाषा (Wiring पर आधारित), और Arduino सॉफ़्टवेयर (IDE) का उपयोग करते हैं।

Arduino Board एक छोटा-सा CPU है, जिसमें एक चिप लगी होती है, जिसे हम Micro-controller या MCU कहते हैं । इस चिप में हम अपना प्रोग्राम Upload कर इसकी Pins को इच्छानुसार Input या Output में set कर मनचाही डिवाइस कनेक्ट कर उसे Control या Hack करके मनचाही चीजें या Project बना सकते हैं ।

Arduino Board के कई Variants आते हैं, जैसे—Arduino UNO, Arduino nano, Arduino mega, Arduino pro mini आदि । इनमें से सबसे लोकप्रिय Arduino UNO है।

8.11.1 Arduino UNO

Arduino UNO एक छोटा-सा Electronic Board है, जो आपके हाथ में बडे आसानी से फिट हो जाएगा। इस Board में ATmega328P Chip लगी होती है, जिसकी क्षमता 32 KB होती है ।

Arduino Uno ATmega328 की विशेषताएँ निम्नलिखित है—

- इसकी ऑपरेटिंग वोल्टेज 5V होती है।
- इसकी अनुशंसित इनपुट वोल्टेज (recommended input voltage) 7V से 12V तक होता है।
- इसकी इनपुट वोल्टेज 6V से 20V तक होता है।
- इसमें डिजिटल इनपुट / आउटपुट पिन 14 होती हैं।
- इसमें एनालॉग i / p पिन 6 हैं।
- इसमें प्रत्येक इनपुट / आउटपुट पिन के लिए DC धारा 40 mA है।

- इसमें 3.3V पिन के लिए DC करंट 50 mA होता है।
- फ्लैश मेमोरी 32 KB है।
- इसमें SRAM 2 KB होता है।
- इसमें EEPROM 1 KB होता है।
- इसमें क्लॉक स्पीड 16 मेगाहर्ट्ज होती है।

8.11.2 Arduino UNO बोर्ड के Component

Arduino UNO बोर्ड के प्रमुख Component निम्नलिखित हैं—

USB कनेक्टर, पॉवर पोर्ट, माइक्रोकंट्रोलर, एनालॉग इनपुट पिन, डिजिटल पिन, रीसेट स्विच, क्रिस्टल ऑसिलेटर, USB इंटरफ़ेस चिप, TX/RX LED। अब प्रत्येक घटक पर करीब से नज़र डालते हैं।

1. यूएसबी कनेक्टर (USB Connector)

पहला Component USB कनेक्टर है, यह एक प्रिंटर USB पोर्ट है, जिसका उपयोग Arduino IDE से एक प्रोग्राम को Arduino बोर्ड पर लोड करने के लिए किया जाता है। Laptop/PC से इस बोर्ड को इस पोर्ट के माध्यम से 500 mA, 5 वोल्ट पर भी संचालित किया जा सकता है।

2. पावर पोर्ट (Power ports) Barrel Connector

Arduino बोर्ड को AC से DC एडॉप्टर या बैटरी के माध्यम से चलाया जा सकता है। इस पॉवर पोर्ट में 2.1 मिलीमीटर Center-positive barrel jack को कनेक्ट किया जा सकता है। Arduino UNO बोर्ड 5 वोल्ट पर संचालित होता है, लेकिन इसे अधिकतम 20 वोल्ट पर चलाया जा सकता है। यदि हम इस बोर्ड को High वोल्टेज देते है, तो voltage regulator, बोर्ड को जलने से बचाता है।

3. Power Pins

Vin पिन, Arduino बोर्ड पर लगे Voltage regulator से जुड़ा होता है, जब भी हम इस बोर्ड पर 7 से 12 वोल्ट Apply करते हैं, तो यह Voltage regulator, Arduino UNO द्वारा उपयोग किए जाने वाले स्थिर 5V में परिवर्तित कर देता है। Arduino बोर्ड को चालू करने के लिए Vin पिन पर 9 volt की बैटरी के पॉजिटिव टर्मिनल को और GND पिन को नेगेटिव पिन से जोड़ सकते हैं।

यदि आपके पास 5 Volts का एक बाहरी स्रोत है, तो आप इसे सीधे Arduino बोर्ड के 5V पिन से जोड़ सकते हैं। 5V पिन पर इनपुट 5.5V वोल्ट से अधिक नहीं होनी चाहिए, नहीं तो आपका बोर्ड जल जाएगा। क्योंकि 5V पिन Voltage regulator को बायपास करता है। यह 5 volt 500ma की Power supply भी देता है, जिससे आप बहरी उपकरणो को power supply दे सकते हैं।

आप 3.3V पिन का उपयोग पॉवर सेंसर और मॉड्यूल के लिए कर सकते हैं, जिन्हें 3.3V 150ma पावर की आवश्यकता होती है।

GND पिन, Arduino Uno में, आप 5 GND पिन हैं, जो सभी आपस में जुड़ी हुई हैं। GND पिन का उपयोग इलैक्ट्रिकल सर्किट को बंद करने के लिए किया जाता है और आपके पूरे सर्किट में एक सामान्य लॉजिक रेफरेंस स्तर प्रदान करता है। हमेशा यह सुनिश्चित करें कि सभी GND (Arduino, बाह्य उपकरण) एक-दूसरे से जुड़े हुए हैं और सबका ग्राउंड (GND) एक ही है।

4. माइक्रोकंट्रोलर (Micro-controller)

अब हम Arduino बोर्ड पर लगे एक बहुत-ही महत्वपूर्ण Component पर नज़र डालते हैं, जो Atmega328p माइक्रोकंट्रोलर है, यह 28 पिन के साथ सबसे प्रमुख रूप से दिखाई देने वाली काली आयताकार चिप है। इसे Arduino

का मस्तिष्क कह सकते हैं। UNO बोर्ड पर उपयोग किया जाने वाला यह माइक्रोकंट्रोलर Atmega328p, Atmel (एक प्रमुख माइक्रोकंट्रोलर निर्माता) द्वारा निर्मित है।

Atmega328p में निम्नलिखित विशेषता हैं—

- 32 kB की फ्लैश मेमोरी जिसे Arduino IDE से लोड किए गए प्रोग्राम को यहां संग्रहीत किया जाता है।
- 2 kB का RAM, जो रनटाइम मेमोरी है।

5. CPU

यह सब कुछ नियंत्रित करता है, जो डिवाइस के अंदर चलता है। यह फ्लैश मेमोरी से प्रोग्राम निर्देशों को प्राप्त करता है और इसे RAM की मदद से चलाता है।

1 kB की EEPROM (Electrically Erasable Programmable Read-Only Memory) होती है जो एक non-volatile memory है, यह डिवाइस को पुनरारंभ करने और रीसेट करने के बाद भी डाटा रखता है।

Atmega328p बूटलोडर के साथ पहले से ही Program होता है। बिना किसी बाहरी हार्डवेयर प्रोग्रामर का उपयोग किए बिना, यह आपको सीधे डिवाइस में एक नया Arduino प्रोग्राम अपलोड करने की अनुमति देता है। जिससे Arduino UNO बोर्ड का उपयोग करना और भी आसान बन जाता है।

6. एनालॉग इनपुट पिन (Analog Input)

अगला एनालॉग इनपुट पिन है, Arduino UNO बोर्ड में 'एनालॉग 0 से 5' (A0 से A5) 6 एनालॉग इनपुट पिन हैं। ये पिन एक एनालॉग सेंसर जैसे तापमान सेंसर के सिग्नल को पढ़ सकते हैं और इसे सिस्टम समझ के लिए डिजिटल रूप में बदल सकते हैं।

ये पिन केवल वोल्टेज को मापते हैं न कि करंट को क्योंकि इनमें बहुत अधिक आंतरिक प्रतिरोध होता है। अत: इन पिनों के माध्यम से केवल थोड़ी मात्रा में करंट प्रवाहित होता है। यद्यपि, इन पिनों को डिफ़ॉल्ट रूप से एनालॉग के रूप में दर्शाया गया है, फिर भी इन पिनों का उपयोग डिजिटल इनपुट या आउटपुट के लिए भी किया जा सकता है।

7. डिजिटल पिन (Digital Pin)

अब डिजिटल पिन को देखते हैं, इन पिन को 'डिजिटल 0 से 13 तक' (D0 से D13) दर्शाया गया है। इन पिनों को इनपुट या आउटपुट पिन के रूप में उपयोग किया जा सकता है। आउटपुट के रूप में उपयोग किए जाने पर, ये पिन इससे जुड़ने उपकरणो के लिए बिजली आपूर्ति स्रोत के रूप में कार्य करती हैं और जब इन्हें इनपुट पिन के रूप में उपयोग किया जाता है, तो ये उनसे जुड़े उपकरणो से प्राप्त संकेतों को पढ़ती हैं।

जब डिजिटल पिन का उपयोग आउटपुट पिन के रूप में किया जाता है, तो ये 5 वोल्ट पर 40 मिलीमीटर की आपूर्ति करते हैं, जो कि एक एलईडी को प्रकाश देने के लिए पर्याप्त से अधिक है।

कुछ डिजिटल पिन, पिन नंबर (पिन नंबर 3, 5, 6, 9, 10 और 11) के बगल में Tilde (~) प्रतीक के साथ दर्शाई गई हैं। ये पिन सामान्य डिजिटल पिन के रूप में कार्य करती हैं, लेकिन Pulse Width Modulation (PWM) के लिए भी उपयोग की जा सकती हैं, जो AnalogWrite फ़ंक्शन के साथ 8-बिट PWM आउटपुट प्रदान करती है।

8. रीसेट स्विच (Reset Switch)

अगला रीसेट स्विच है, जब यह स्विच क्लिक किया जाता है, तो यह माइक्रोकंट्रोलर के रीसेट पिन पर एक तार्किक पल्स भेजता है, और अब फिर से शुरू से Program को चलाता है। यदि आपका कोड दोहराता नहीं है और आप इसका कई बार परीक्षण करना चाहते हैं, तो यह बहुत उपयोगी हो सकता है।

9. क्रिस्टल ऑसिलेटर (Crystal Occilator)

इस बोर्ड में 16MHz का क्रिस्टल ऑसिलेटर लगा होता है, जो 1 सेकंड में 16 मिलियन बार कम्पन करता है, जिसे फ्रीक्वेंसी कहते हैं और इसके प्रत्येक कम्पन पर, माइक्रोकंट्रोलर एक ऑपरेशन करता है, उदाहरण के लिए, जोड़, घटा आदि।

10. USB इंटरफ़ेस चिप (USB Interface Chip)

अब हम USB इंटरफ़ेस चिप ATmega16U2 को देखेंगे, यह एक सिग्नल ट्रांसलेटर है। यह USB से प्राप्त संकेतों को Arduino UNO बोर्ड के समझने लायक संकेतों में परिवर्तित करता है।

11. Tx/Rx इंडिकेटर (Tx/Rx Indicator)

अंतिम एक TX RX संकेतक है, TX LED सन्देश भेजने और RX LED सन्देश प्राप्त होने पर जलता है, जब भी UNO बोर्ड डाटा प्रसारित या प्राप्त करता है तो ये LED बार-बार जलते बुझते हैं।

8.12 अरूड्यूनों प्रोग्रामिंग भाषा (Arduino Programming Language)

Arduino Programming पूरी तरह प्रोग्रामिंग C Based है, यदि आपने पहले कभी C language पर काम किया है, तब यह आपको बिलकुल भी कठिन नहीं लगेगी आप आसानी से Arduino Programming कर सकते है।

8.12.1 ARDUINO IDE

ARDUINO की कोई अलग प्रोग्रामिंग लैंग्वेज नहीं है, बल्कि ARDUINO को हम C प्रोग्रामिंग से कोड करते है और उसे ARDUINO C के नाम से जाना जाता है। ARDUINO C में लिखे गए कोड को SKETCH कहते हैं।

ARDUINO IDE का अर्थ ARDUINO Integrated Development Environment है। IDE के उपयोग से हम ARDUINO को Code करते है और IDE की मदद से ही ARDUINO में कोड Dump (अपलोड) करते है।

आप ARDUINO IDE को www.arduino.cc से डाउनलोड कर सकते हैं।

8.12.2 ARDUINO IDE को डाउनलोड करने के चरण

- सबसे पहले आप arduino.cc पर जाइए।
- फिर आप Menu bar में software>downloads पर क्लिक करे। क्लिक करने पर आपको कुछ एसी स्क्रीन मिलेगी.
- ARDUINO community अब आपको ARDUINO को कोड करने के लिए दो विकल्प दे रही है।
  - ARDUINO web editor—इससे आप बिना कुछ डाउनलोड किए अपने ARDUINO को इन्टरनेट से कोड कर सकते हैं।
  - ARDUINO IDE—इस IDE के माध्यम से कोड करेंगे। IDE के बारे में अधिक जानकारी आगे इस article में दी गई है।
- Downloads के पेज में नीचे जाने पर आपको डाउनलोड करने का ऑप्शन मिलेगा। यहाँ पर आपको Windows, Mac और Linux तीनो OS के लिए डाउनलोड का ऑप्शन मिलेगा।
- किसी एक ऑप्शन पर क्लिक करने के बाद अगले पेज पर आप डाउनलोड कर पाएँगे। हमे यह तो पता है की ARDUINO open source कम्युनिटी है, इसलिए आप उनको DONATE करके उनको प्रोत्साहित कर सकते है।

आपने इन सभी चरणों का सही तरीके से पालन किया होगा, तो आपने ARDUINO IDE सही से डाउनलोड कर लिया है। चलिए तो अब इसको गौर से देखते और समझते है। इसके Menu Bar में हमे *files, edit, sketch, tools* और *help* आदि *TABS* मिलेगे।

ARDUINO project को Save, rename, Open File Menu से कर पाएंगे। File के अन्दर आपको Examples का ऑप्शन मिलेगा। जिसमे आपको ARDUINO की तरफ से ही ढेर सारे SKETCH पहले से ही मिलेंगे। जिनको आप लेकर आसानी से कुछ ARDUINO project कर पाएँगे। Edit option में आपको Copy व Paste के ऑप्शन मिलेंगे। आपने जो कोड लिखा है उसे आप compile और upload, sketch टैब से कर पाएंगे।

अब ARDUINO का सबसे मुख्य टैब, Tools, की बात करते है। इसमें हम सबसे पहले BOARDS के बारे में बात करते हैं। ARDUINO IDE से आप केवल ARDUINO को ही नहीं बल्कि nodeMCU, adaFruit के BOARDS और दूसरे कोड कर सकते हैं। आप अपने Tools > BoardsceW जाइए और अपना बोर्ड Select कर लीजिए। सही बोर्ड Select करना एक Important STEP है।

अब आप Tools > Board में गए लेकिन आपको आपका डेवलपमेंट बोर्ड नहीं मिला। तो फिर आप Tools >Board > Board Manager में जाइए और आपका बोर्ड Search करके उसकी फाइल्स डाउनलोड कर लें।

दूसरी बात जिसका ध्यान रखना है, वो है Port। सही Port select करना महत्वपूर्ण है, क्योंकि आपके PC/ LAPTOP के साथ केवल ARDUINO ही नहीं बल्कि और कई सारी DEVICES जुडी होंगी।

Setup

1. Power the board by connecting it to a PC via USB cable.
2. Launch the Arduino IDE.
3. Set the board type and the port for the board.
4. TOOLS -> BOARD -> select your board.
5. TOOLS -> PORT -> select your port.

Supported Datatype

Arduino supports the following data types :

1. Void Long
2. Int
3. Char
4. Unsigned char
5. Unsigned int
6. Unsigned long
7. Float
8. String
9. Array String
10. char array

Arduino Function Libraries

1. Input/Output Functions :

(i) The arduino pins can be configured to act as input or output pins using the pinMode() function :

```
Void setup ()
{
pinMode (pin , mode);
}
```

Pin- pin number on the Arduino board Mode- INPUT/OUTPU

digitalWrite() : Writes a HIGH or LOW value to a digital pin.

- analogRead() : Reads from the analog input pin *i.e.*, voltage applied across the pin.

- Character functions such as isdigit(), isalpha(), isalnum(), isxdigit(), islower(), isupper(), isspace() return 1(true) or 0(false)

- Delay( ) function is one of the most common time manipulation function used to provide a delay of specified time. It accepts integer value (time in miliseconds)

Control Statement :

- If statement
- if(condition) {

Statements if the
condition is true ;
}

- If...Else statement
- if(condition) {

Statements if the
condition is true;
}
else{
Statements if the
condition is false;
}

4

- If.....Elseif.....Else
- if (condition1){

Statements if the
condition1 is true;
}
else if (condition2){
Statements if the
condition1 is false
and condition2 is true;
}
else{
Statements if both the
conditions are false;

}

- Switch Case
- Switch(choice)

{
case opt1: statement_1;break;
case opt2: statement_2;break;
case opt3: statement_3;break;

•
•
•

case default: statement_default; break;
}

- Conditional Operator.
- Val=(condition)?(Statement1): (Statement2)

Loops

- For loop
- . for(initialization; condition; increment){

Statement till the condition is true;
}

- While loop
- while(condition){

Statement till the condition is true;
}

- Do... While loop
- do{

Statement till the condition is true;
}while(condition);

- Nested loop: Calling a loop inside another loop

- Infinite loop: Condition of the loop is always true, the loop will never terminate.

प्रश्नावली

1. इंटरनेट ऑफ थिंग्स (IoT) से आप क्या समझते हैं?
2. इंटरनेट ऑफ थिंग्स (Internet of Things) के अनुप्रयोगों की व्याख्या कीजिए।
3. Internet of things का आर्किटेक्चर बनाकर व्याख्या कीजिए।
4. Internet of things के प्रोटोकॉल लिखिए।
5. इंटरनेट ऑफ थिंग्स के विभिन्न फंक्शनल ब्लॉक का वर्णन कीजिए।
6. इंटरनेट ऑफ थिंग्स के अभिलक्षण लिखिए।
7. ऑरडिनो (Aurduino) IDE के बारे में बताइए।
8. IoT की विभिन्न समस्याओं की व्याख्या कीजिए।
9. मशीन-टू-मशीन (Machine-to-machine) कम्यूनिकेशन क्या होता है?
10. IoT में विभिन्न प्रकार के wired और wireless का वर्णन कीजिए।
11. IoT प्रोटोकॉल को Standardize (मानकीकृत) करने की आवश्यकता क्यों है?
12. Internet of things security की आवश्यकता बताएँ।
13. IoT security में क्या समस्याएँ हैं?
14. निम्न पर टिप्पणी लिखें—
    (i) Trust for IoT
    (ii) Security and privacy for IoT
    (iii) Physical IoT security.

चित्र 1.3

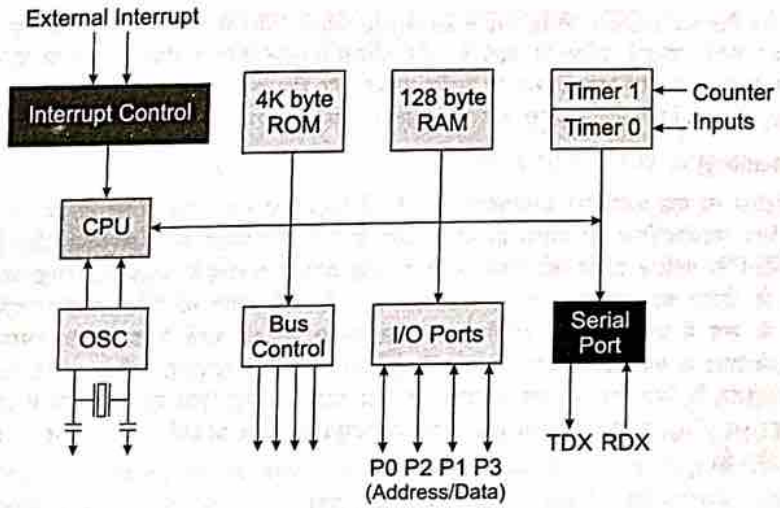
मुख्य रूप से, N-MOS टेक्नोलॉजी का उपयोग करके 8051 माइक्रोकंट्रोलर्स विकसित किए गए थे, लेकिन बैटरी चालित उपकरणों के उपयोग और उनके कम बिजली खपत के लिए इनमें CMOS टेक्नोलॉजी का उपयोग किया जाता है। INTEL ने 8051 माइक्रोकंट्रोलर विकसित किए थे, जिनका प्रचालन सन् 2007 में बंद हो गया था। 20 से अधिक सेमीकंडक्टर निर्माता अभी भी 8051 संगत माइक्रोकंट्रोलर अर्थात् प्रोसेसर का निर्माण कर रहे हैं जो MSC-51 आर्किटेक्चर पर आधारित हैं। विभिन्न निर्माताओं द्वारा निर्मित 8051 माइक्रोकंट्रोलर में से कुछ हैं—Atmel (AT89C51, AT89S51), फिलिप्स (S87C654), STC माइक्रो (STC89C52), Infineon (SAB-C515, XC800), Siemens (SAB-C501), सिलिकॉन लैब्स (C8051) , NXP (NXP700, NXP900), आदि। आधुनिक 8051 माइक्रोकंट्रोलर्स की अधिकांश संख्या सिलिकॉन आईपी कोर (Intellectual Property Cores) हैं, लेकिन असतत 8051 माइक्रोकंट्रोलर आईसी भी उपलब्ध हैं। उनकी कम बिजली की खपत, छोटे आकार और सरल वास्तुकला के कारण, उन्नत एआरएम (ARM) आर्किटेक्चर आधारित MCUs के बजाय FPGAs (फील्ड प्रोग्रामेबल गेट ऐरे) और SOCs (सिस्टम ऑन चिप) में 8051 IP Cores का उपयोग किया जाता है।

1.6 8051 माइक्रोकंट्रोलर का आर्किटेक्चर (8051 Microcontroller Architecture)

जब भी हम किसी नए उपकरण, जैसे—टीवी या वॉशिंग मशीन पर कार्य करना शुरू करते हैं, तो हम उपकरण की क्षमता के बारे में समझकर कार्य करना शुरू करते हैं। जैसे हम वॉशिंग मशीन की विभिन्न विशेषताओं; जैसे—मोटर आरपीएम, लोड क्षमता और पॉवर की खपत को समझने की कोशिश करते हैं। यह हम पर भी लागू होता है, यानी जब 8051 माइक्रोकंट्रोलर से शुरू होता है, तो यह सबसे अच्छा होगा यदि हमने 8051 माइक्रोकंट्रोलर के आंतरिक हार्डवेयर डिजाइन को सीखना शुरू किया, जिसे 8051 माइक्रोकंट्रोलर आर्किटेक्चर भी कहते हैं।

8051 माइक्रोकंट्रोलर एक 8-बिट माइक्रोकंट्रोलर है यानी यह 8-बिट के डाटा को पढ़, लिख और प्रोसेस कर सकता है। Atmel, NXP, TI जैसे निर्माताओं का एक समूह है, जो 8051 माइक्रोकंट्रोलर के अपने संस्करणों का निर्माण करते हैं। निर्माता की परवाह किए बिना, आंतरिक हार्डवेयर डिजाइन यानी 8051 माइक्रोकंट्रोलर आर्किटेक्चर एक ही रहता है। दिया गया चित्र 1.4 एक ब्लॉक आरेख शैली में 8051 माइक्रोकंट्रोलर आर्किटेक्चर को दर्शाता है।

8051 माइक्रोकंट्रोलर आर्किटेक्चर के ब्लॉक आरेख से पता चलता है, कि 8051 माइक्रोकंट्रोलर में सीपीयू, रैम (एसएफआर और डाटा मेमोरी), फ्लैश (EEPROM), Input/Output Ports होते हैं और बाह्य उपकरणों के बीच संचार के लिए नियंत्रण लॉजिक होते हैं। 8051 माइक्रोकंट्रोलर में ये सभी अलग-अलग पेरिफेरेल्स 8-बिट डाटा बस के माध्यम से एक-दूसरे के साथ कार्य करते है, जिसे आंतरिक डाटा बस भी कहा जाता है।



चित्र 1.4

8051 माइक्रोकंट्रोलर आर्किटेक्चर के आंतरिक घटक निम्नलिखित हैं—

1. Central Processor Unit (CPU)

जैसा कि हम जानते हैं, कि सीपीयू माइक्रोकंट्रोलर के किसी भी प्रोसेसिंग डिवाइस का मस्तिष्क होता है। यह माइक्रोकंट्रोलर इकाइयों पर किए जाने वाले सभी ऑपरेशनों को नियंत्रित करता है। सीपीयू के कार्य पर उपयोगकर्ता का कोई नियंत्रण नहीं होता है। यह ROM मेमोरी में लिखे प्रोग्राम को पढ़ता है और उन्हें निष्पादित करता है और उस एप्लिकेशन के अपेक्षित कार्य को करता है।

2. Interrupts (व्यवधान)

जैसा कि इसके नाम से ज्ञात होता है, Interrupt एक Sub-routine call है, जो माइक्रोकंट्रोलर्स के मुख्य संचालन या कार्य में बाधा डालती है और इसके कारण किसी अन्य कार्यक्रम को निष्पादित करती है, जो ऑपरेशन के समय अधिक महत्वपूर्ण होते हैं। Interrupt की सुविधा बहुत उपयोगी है, क्योंकि यह आपातकालीन कार्यक्रम के संचालन के में मदद करता है। एक व्यवधान हमें चल रहे संचालन को रोकने के लिए, एक Sub-routine को निष्पादित करने और फिर दूसरे प्रकार के संचालन के लिए फिर से शुरू करने के लिए एक तंत्र देता है।

माइक्रोकंट्रोलर 8051 को इस प्रकार से कॉन्फ़िगर किया जा सकता है, कि यह व्यवधान की घटना पर मुख्य कार्यक्रम को अस्थायी रूप से समाप्त या रोक देता है। जब एक सबरूटीन पूरा हो जाता है, तो मुख्य कार्यक्रम का निष्पादन शुरू होता है। 8051 माइक्रोकंट्रोलर में मुख्य रूप से पांच बाधा स्रोत होते हैं, जो कि निम्नलिखित होते हैं—

INT0
TF0
INT1
TF1
R1/T1

इनमें (INT0) INT और (INT1) बाहरी व्यवधान हैं, जो नकारात्मक वृद्धि को ट्रिगर या निम्न स्तर पर ट्रिगर कर सकते हैं। जब इन सभी Interrupt को सक्रिय किया जाता है, तो सीरियल Interrupt को छोड़कर संबंधित पल्स को