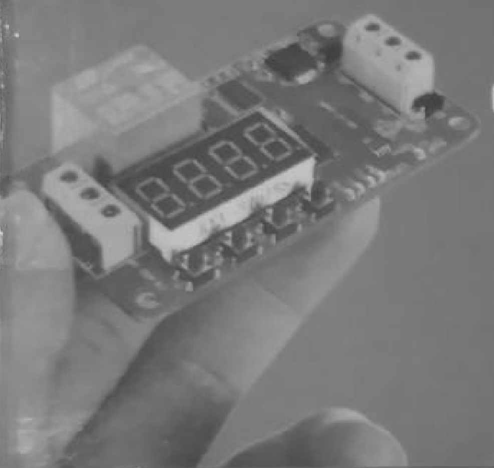
सार्थक

प्रविधिक शिक्षा परिषद् उ०प्र० द्वारा स्वीकृत
नवीनतम "N.S.Q.F." पाठ्यक्रमानुसार

# पी एल सी, माइक्रोकन्ट्रोलर एवं SCADA

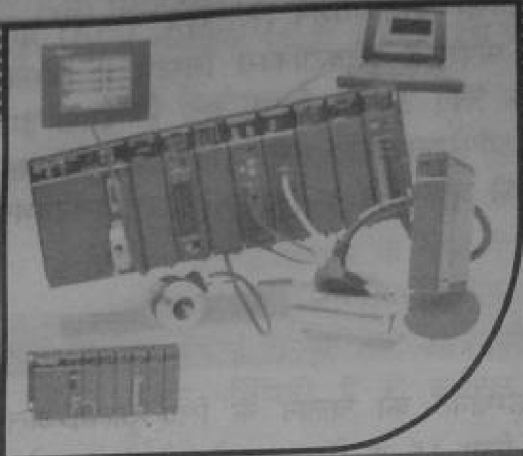
## PLC, Microcontroller & SCADA



**Vth Semester**

कुमार • कुमार

**JAI PRAKASH NATH PUBLICATIONS**
MEERUT



# 1

# पीएलसी का परिचय
# (Introduction to P.L.C.)

## प्रोग्रामेबल लॉजिक कन्ट्रोलर (PLC) तथा तकनीकी परिभाषा (P.L.C. and Technical Definition)

### प्रोग्रामेबल लॉजिक कन्ट्रोलर (Programmable Logic Controller)

प्रोग्रामेबल लॉजिक कन्ट्रोलर एक इलेक्ट्रॉनिक डिवाइस है जो डिजिटल सिग्नल पर ऑपरेट होती है। प्रोग्रामेबल लॉजिक कन्ट्रोलर को कम्प्यूटर की सहायता से ऑपरेट किया जाता है। PLC, मशीन तथा मानव के बीच इन्टरफेसिंग का कार्य करती है। इसकी सहायता से किसी भी मशीन को ऑटोमेटिक कन्ट्रोल किया जाता है तथा विभिन्न प्रकार के ऑपरेशन परफॉर्म किये जाते हैं। PLC द्वारा विभिन्न प्रकार की ऑटोमेशन एवं मशीन क्रियाओं को कन्ट्रोल किया जाता है। इसकी सहायता से विभिन्न प्रकार के इन्स्ट्रक्शनों को मैमोरी में स्टोर किया जाता है तथा सम्बन्धित ऑपरेशन परफॉर्म होने पर उन्हें उपयोग में लाया जाता है।

PLC सामान्यत: एक कम्प्यूटर है जिसका उपयोग इन्डस्ट्रियल ऑपरेशनों को कन्ट्रोल करने में किया जाता है। इसके सभी फंक्शन कम्प्यूटर के समान ही होते हैं जो हार्डवेयर तथा सॉफ्टवेयर पर सम्पन्न होते हैं। PLC में सेन्ट्रल प्रोसेसिंग यूनिट, मैमोरी, इनपुट आउटपुट पोर्ट, टाइमर पर काउन्टर, एनालॉग-टू-डिजिटल कन्वर्टर, डिजिटल-टू-एनालॉग कन्वर्टर, सीरियल पोर्ट, इन्ट्रप्ट लॉजिक, ऑसीलेटर आदि ऑपरेशन परफॉर्म किये जाते हैं। इसे प्रकार PLC में विभिन्न प्रकार के इन्टीग्रेटेड सर्किटों का उपयोग किया जाता है जो इससे सम्बन्धित विभिन्न क्रियाओं को ऑपरेट करता है। प्रोग्रामेबल लॉजिक कन्ट्रोलर सॉलिड-स्टेट डिजिटल लॉजिक एलीमेन्ट का एक मिला-जुला रूप है जो लॉजिकल निर्णय लेने एवं आउटपुट प्रदान करने के लिए डिजाइन किया जाता है। PLC का उपयोग निर्माणकारी इकाइयों के यंत्रों एवं उनकी मशीनरी को ऑपरेट तथा कन्ट्रोल करने के लिये किया जाता है। आजकल इसका उपयोग मुख्य रूप से रिमोट कन्ट्रोल, टेलीफोन बिल, प्रिन्टिंग मशनी, ऑटोमेटिक पावर रेगुलेटर ऑटोमेटिक एवं सेमीऑटोमेटिक मशीनों, माइक्रोवेव ओवन, ऑटोमोबाइल तथा मेजरमेन्ट यंत्रों से सम्बन्धित डिवाइसों में किया जाता है। इस प्रकार PLC के उपयोग से किसी भी कार्यक्षेत्र के आकार को छोटा किया जा सकता है तथा इसकी कार्यक्षमता के समय को कम किया जा सकता है एवं उत्पाद को बढ़ाया जा सकता है। प्रोग्रामेबल लॉजिक कन्ट्रोलर का उपयोग उत्पाद के क्रमों को कन्ट्रोल करने के लिए किया जाता है, जिसे सिक्वेन्शियल कन्ट्रोलर्स के नाम से भी जाना जाता है।

उदाहरण के लिए, एक कार बनाने वाली कम्पनी के द्वारा किसी कार का निर्माण करने के लिए सर्वप्रथम उसका स्ट्रेक्चर तैयार किया जाता है। इसके बाद उसके पार्टस बनाये जाते हैं तथा सभी की एक प्रोसेसिंग की सहायता से जोड़ा जाता है एवं उसकी बॉडी तैयार की जाती है और एक निश्चित कलर दिया जाता है। इस प्रकार यह क्रिया अनेक स्टेप में पूर्ण होती है, जिसे ऑटोमेटिक मैन्यूफैक्चरिंग कहते हैं। इसमें मुख्यत: PLC का उपयोग किया जाता है।

## PLC की तकनीकी परिभाषा (Technical Definition of PLC)

PLC एक टेक्नीकल डिवाइस है जिसे नेशनल इलेक्ट्रिकल मैन्यूफैक्चर एसोसियेशन (NEMA, USA) द्वारा परिभाषित किया गया है। "प्रोग्रामेबल लॉजिक कन्ट्रोलर एक डिजिटल ऑपरेटेड इलेक्ट्रॉनिक्स सिस्टम है जिसकी इन्डस्ट्रियल वातावरण में विभिन्न प्रकार के फंक्शनों को परफॉर्म करने के लिए उपयोग किया जाता है। इसके द्वारा विशिष्ट फंक्शन, जैसे–लॉजिक सिक्वेसिंग, टाइमिंग, काउन्टिग, अर्थमैटिक ऑपरेशन आदि सम्पन्न किये जाते हैं। इसमें एनालॉग इनपुट तथा आउटपुट के द्वारा विभिन्न प्रकार की मशीनों एवं उनकी प्रक्रियाओं के लिए महत्त्वपूर्ण इन्स्ट्रक्शनों का उपयोग किया जाता है जो मैमोरी में स्टोर होते हैं।"

## P.L.C. के लाभ (Advantages of P.L.C.)

**(i) फ्लैक्सिबिलिटी** (Flexibility)—पहले प्रत्येक इलेक्ट्रॉनिक कम्पोनेट को चलाने के लिए अलग-अलग कन्ट्रोलर की आवश्यकता होती थी। इस प्रकार 15 मशीनों को चलाने के लिए 15 कन्ट्रोलर उपयोग में लाये जाते थे लेकिन अब 15 मशीनो को एक ही कन्ट्रोलर से ऑपरेट किया जा सकता है।

**(ii) विजुअल ऑपरेशन** (Visual Operation)—PLC की सहायता से किसी ऑपरेशन को CRT तथा मॉनीटर स्क्रीन पर सीधे देखा जा सकता है। इस प्रकार से किसी ऑपरेशन के दौरान अगर कोई फॉल्ट होता है तो प्रोसेस प्रक्रिया को वहीं पर रोका जा सकता है।

**(iii) ऑपरेशन की स्पीड** (Speed of Operation)—PLC प्रोग्रामिंग की स्पीड बहुत अधिक होती है। इसमें रिले द्वारा ऑपरेशन को इतनी जल्दी ऑपरेट नहीं किया जा सकता है। PLC कुछ ही समय में ऑपरेशन को परफॉर्म कर देती है जिसका ऑपरेशनल टाइम कुछ मिली सेकण्ड होता है।

**(iv) लैडर अथवा बूलियन प्रोग्रामिंग मैथर्ड** (Ladder or Boolean Programming Method)—लैडर अथवा बूलियन प्रोग्रामिंग को इलेक्ट्रीशियन अथवा टेक्नीशियन द्वारा लैडर मोड में सम्पन्न की जाती है। इस प्रकार से किसी लैडर अथवा बूलियन प्रोग्रामर द्वारा यह प्रोग्रामिंग सम्पन्न की जाती है।

**(v) फास्टर रिस्पॉन्स टाइम** (Faster Response Time)—PLC को अधिक स्पीड वाली एक रीयल टाइम एप्लीकेशन के लिए निर्मित किया जाता है। प्रोग्रामेबल कन्ट्रोलर्स रीयल टाइम में ऑपरेट होते हैं। इस प्रकार से PLC किसी ऑपरेशन को सिर्फ कमाण्ड देने पर ही ऑपरेट कर देता है, जिससे किसी भी प्रोसेस की गति को बढ़ाया जा सकता है।

**(vi) मिनिमम मेन्टीनेन्स** (Minimum Maintenance)—PLC के द्वारा ऑपरेट किये जाने वाले यंत्रों को कम मेन्टीनेन्स की आवश्यकता होती है।

**(vii) ट्रबलशूट के लिए आसान** (Easier to Troubleshoot)—PLC में रेजीडेन्ट डायग्नोस्टिक्स एवं ओवरराइड फंक्शन्स होते हैं जो उपयोगकर्त्ता को सॉफ्टवेयर एवं हार्डवेयर की समस्याओं को आसानी से ट्रेस एवं सही करने की अनुमति प्रदान करते हैं, जिससे कि उपयोगकर्त्ता उस समस्या को आसानी से हल कर सके।

**(viii) पाइलट रनिंग** (Pilot Running)—PLC प्रोग्राम तथा सर्किट पहले से टेस्ट किया हुआ होता है जिसे किसी ऑफिस या लैब में रन किया जाता है। इसके प्रोग्राम तथा सर्किट में उत्पादन के अनुसार कुछ बदलाव किया जाता है।

**(ix) लोअर कॉस्ट** (Lower Cost)—PLC में उपलब्ध विभिन्न फंक्शन, जैसे—रिले, टाइमर, काउन्टर तथा सिक्वेन्सर को बहुत ही कम कॉस्ट में प्राप्त किया जा सकता है।

**(x) डॉक्यूमेन्टेशन** (Documentation)—PLC के डॉक्यूमेन्टेशन में किसी सर्किट के प्रिंटआउट को कुछ ही समय में प्रापत किया जाता है तथा उसे आसानी से उपयोग बनाया जाता है। इसमें PLC बुकलेट का उपयोग किया जाता है।

**(xi) कम्युनिकेशन केपेबिलिटी** (Communication Capability)—PLC विभिन्न प्रकार के ऑपरेशन परफॉर्म करने के लिए विभिन्न प्रकार के फंक्शन्स, जैसे—सुपरवाइजरी कन्ट्रोल, डाटा-गेदरिंग, डिवाइसेज पैरामीटर, डाउनलोडिंग तथा अपलोडिंग आदि को परफॉर्म करता है तथा उनसे सम्पर्क रखता है।



**(xii) सिक्योरिटी** (Security)—PLC प्रोग्राम में परिवर्तन तब तक नहीं किया जा सकता है जब तक कि वह उचित प्रकार से अनलॉक नहीं हो जाता है। इस प्रकार से PLC के डाटा को चोरी करने से रोका जा सकता है।

## P.L.C. की विशेषताएँ

(i) यह यूजर के लिए उपयोगी प्रोग्रामेबल क्षेत्र है, जो यूजर को किसी भी प्रोग्राम को मैन्यूफैक्चरिंग के अनुसार मॉडीफाइ करने का आदेश देता है।

(ii) यह विभिन्न प्रोग्रामों के फंक्शनों को रखता है, जो लैस्ट लॉजिक, टाइमिंग, काउन्टिग तथा मैमोरी आदि फंक्शनों को कन्ट्रोल, ओरियेंटिड प्रोग्रामिंग लैंग्वेज द्वारा एक्सेस करता है।

(iii) यह मैमोरी, इनपुट तथा आउटपुट को स्कैन करने का पहले से निर्णय ले लेता है। इस प्रकार से PLC कठिन परिस्थिति में भी कन्ट्रोल इंजीनियर को मशीन ऑपरेशन तथा प्रोग्राम की प्रोसेस में मदद करती है।

(iv) यह त्रुटि तथा वायरस को चैक करती है। इस प्रकार से PLC मैमोरी, प्रोसेसर तथा I/O सिस्टम को एक निश्चित समय में टेस्ट करती है तथा इसके मशीन और सिस्टम को एक एक्जीक्यूशन प्रोग्राम में शामिल करती है।

(v) PLC के द्वारा मॉनीटरिंग क्षमता को इन्डीकेशन लेम्प व इनपुट तथा आउटपुट द्वारा दर्शाया जाता है, फिर इसे एक्जीक्यूशन प्रोग्राम स्टेटस द्वारा दर्शाया जाता है।

(vi) यह टेम्प्रेचर, आर्द्रता, वाइब्रेशन तथा नॉइस को प्राप्त करती है तथा उन्हें वातावरण से प्रभावित नहीं होने देती।

(vii) इस प्रकार से PLC को किसी प्रभावी कन्ट्रोल टास्क के लिए बनाया जाता है जो किसी निश्चित कार्य में भी उपयोग की जाती है।

## P.L.C. तथा कम्प्यूटर में अन्तर (Difference between Computer and P.L.C.)

| क्र० सं० | पी०एल०सी० (PLC) | कम्प्यूटर (Computer) |
|---|---|---|
| (i) | PLC एक इलेक्ट्रॉनिक डिवाइस है जो डिजिटल सिग्नल पर ऑपरेट होती है। | कम्प्यूटर एक माइक्रोप्रोसेसर आधारित डिवाइस है जो गणितीय क्रियाओं को सम्पन्न करती है। |
| (ii) | PLC सभी क्रियाओं को ऑन लाइन स्क्रीन पर दर्शाता है। | इसमें सभी क्रियायें इसके आन्तरिक भाग में सम्पन्न होती हैं। |
| (iii) | PLC के द्वारा किसी भी ऑपरेशन को कुछ ही मिली सेकण्ड में कार्यान्वित किया जाता है। इसमें स्कैन का समय 1/60 सेकण्ड से भी कम होता है। | इसमें सम्भवत: जटिल प्रोग्राम सम्पन्न होते है तथा टाइमिंग ज्यादा क्रिटिकल नहीं होती है। |
| (iv) | इसको किसी विशिष्ट उद्देश्य के लिए उपयोग में लाया जाता है तथा इसमें एक ही प्रोग्राम की पुनरावृत्ति होती है। | इसमें सभी फंक्शनों के लिए अलग-अलग प्रोग्राम होते हैं। |
| (v) | इसमें पैरीफेरल डिवाइस का उपयोग नहीं किया जाता है। | इसमें विभिन्न प्रकार की पैरीफेरल डिवाइसों का उपयोग किया जाता है। |
| (vi) | इसमें सभी प्रोग्राम लैडर प्रोग्रामिंग की सहायता से लिखे जाते हैं। | इसमें सभी प्रोग्राम असेम्बली भाषा में लिखे जाते हैं। |

## PLC की तकनीकी परिभाषा (Technical Definition of PLC)

PLC एक टेक्नीकल डिवाइस है जिसे नेशनल इलेक्ट्रिकल मैन्यूफैक्चर एसोसियेशन (NEMA, USA) द्वारा परिभाषित किया गया है। "प्रोग्रामेबल लॉजिक कन्ट्रोलर एक डिजिटल ऑपरेटेड इलेक्ट्रॉनिक्स सिस्टम है जिसको इन्डस्ट्रियल वातावरण मे विभिन्न प्रकार के फंक्शनों को परफॉर्म करने के लिए उपयोग किया जाता है। इसके द्वारा विशिष्ट फंक्शन, जैसे–लॉजिक सिक्वेसिंग, टाइमिंग, काउन्टिग, अर्थमैटिक ऑपरेशन आदि सम्पन्न किये जाते हैं। इसमें एनालॉग इनपुट तथा आउटपुट के द्वारा विभिन्न प्रकार की मशीनों एवं उनकी प्रक्रियाओं के लिए महत्त्वपूर्ण इन्स्ट्रक्शनों का उपयोग किया जाता है जो मैमोरी में स्टोर होते हैं।"

## P.L.C. के लाभ (Advantages of P.L.C.)

**(i) फ्लैक्सिबिलिटी** (Flexibility)—पहले प्रत्येक इलेक्ट्रॉनिक कम्पोनेट को चलाने के लिए अलग-अलग कन्ट्रोलर की आवश्यकता होती थी। इस प्रकार 15 मशीनों को चलाने के लिए 15 कन्ट्रोलर उपयोग में लाये जाते थे लेकिन अब 15 मशीनों को एक ही कन्ट्रोलर से ऑपरेट किया जा सकता है।

**(ii) विजुअल ऑपरेशन** (Visual Operation)—PLC की सहायता से किसी ऑपरेशन को CRT तथा मॉनीटर स्क्रीन पर सीधे देखा जा सकता है। इस प्रकार से किसी ऑपरेशन के दौरान अगर कोई फॉल्ट होता है तो प्रोसेस प्रक्रिया को वहीं पर रोका जा सकता है।

**(iii) ऑपरेशन की स्पीड** (Speed of Operation)—PLC प्रोग्रामिंग की स्पीड बहुत अधिक होती है। इसमें रिले द्वारा ऑपरेशन को इतनी जल्दी ऑपरेट नहीं किया जा सकता है। PLC कुछ ही समय में ऑपरेशन को परफॉर्म कर देती है जिसका ऑपरेशनल टाइम कुछ मिली सेकण्ड होता है।

**(iv) लैडर अथवा बूलियन प्रोग्रामिंग मैथर्ड** (Ladder or Boolean Programming Method)—लैडर अथवा बूलियन प्रोग्रामिंग को इलेक्ट्रीशियन अथवा टेक्नीशियन द्वारा लैडर मोड में सम्पन्न की जाती है। इस प्रकार से किसी लैडर अथवा बूलियन प्रोग्रामर द्वारा यह प्रोग्रामिंग सम्पन्न की जाती है।

**(v) फास्टर रिस्पॉन्स टाइम** (Faster Response Time)—PLC को अधिक स्पीड वाली एक रीयल टाइम एप्लीकेशन के लिए निर्मित किया जाता है। प्रोग्रामेबल कन्ट्रोलर्स रीयल टाइम में ऑपरेट होते हैं। इस प्रकार से PLC किसी ऑपरेशन को सिर्फ कमाण्ड देने पर ही ऑपरेट कर देता है, जिससे किसी भी प्रोसेस की गति को बढ़ाया जा सकता है।

**(vi) मिनिमम मेन्टीनेन्स** (Minimum Maintenance)—PLC के द्वारा ऑपरेट किये जाने वाले यंत्रों को कम मेन्टीनेन्स की आवश्यकता होती है।

**(vii) ट्रबलशूट के लिए आसान** (Easier to Troubleshoot)—PLC में रेजीडेन्ट डायग्नोस्टिक्स एवं ओवरराइड फंक्शन्स होते हैं जो उपयोगकर्त्ता को सॉफ्टवेयर एवं हार्डवेयर की समस्याओं को आसानी से ट्रेस एवं सही करने की अनुमति प्रदान करते हैं, जिससे कि उपयोगकर्त्ता उस समस्या को आसानी से हल कर सके।

**(viii) पाइलट रनिंग** (Pilot Running)—PLC प्रोग्राम तथा सर्किट पहले से टेस्ट किया हुआ होता है जिसे किसी ऑफिस या लैब में रन किया जाता है। इसके प्रोग्राम तथा सर्किट में उत्पादन के अनुसार कुछ बदलाव किया जाता है।

**(ix) लोअर कॉस्ट** (Lower Cost)—PLC में उपलब्ध विभिन्न फंक्शन, जैसे—रिले, टाइमर, काउन्टर तथा सिक्वेन्सर को बहुत ही कम कॉस्ट में प्राप्त किया जा सकता है।

**(x) डॉक्यूमेन्टेशन** (Documentation)—PLC के डॉक्यूमेन्टेशन में किसी सर्किट के प्रिंटआउट को कुछ ही समय में प्रापत किया जाता है तथा उसे आसानी से उपयोग बनाया जाता है। इसमें PLC बुकलेट का उपयोग किया जाता है।

**(xi) कम्युनिकेशन केपेबिलिटी** (Communication Capability)—PLC विभिन्न प्रकार के ऑपरेशन परफॉर्म करने के लिए विभिन्न प्रकार के फंक्शन्स, जैसे—सुपरवाइजरी कन्ट्रोल, डाटा-गेदरिंग, डिवाइसेज पैरामीटर, डाउनलोडिंग तथा अपलोडिंग आदि को परफार्म करता है तथा उनसे सम्पर्क रखता है।



**(xii) सिक्योरिटी** (Security)—PLC प्रोग्राम में परिवर्तन तब तक नहीं किया जा सकता है जब तक कि वह उचित प्रकार से अनलॉक नहीं हो जाता है। इस प्रकार से PLC के डाटा को चोरी करने से रोका जा सकता है।

## P.L.C. की विशेषताएँ

(i) यह यूजर के लिए उपयोगी प्रोग्रामेबल क्षेत्र है, जो यूजर को किसी भी प्रोग्राम को मैन्यूफैक्चरिंग के अनुसार मॉडीफाइ करने का आदेश देता है।

(ii) यह विभिन्न प्रोग्रामों के फंक्शनों को रखता है, जो लीस्ट लॉजिक, टाइमिंग, काउन्टिग तथा मैमोरी आदि फंक्शनों को कन्ट्रोल, ओरियेंटिड प्रोग्रामिंग लैंग्वेज द्वारा एक्सेस करता है।

(iii) यह मैमोरी, इनपुट तथा आउटपुट को स्कैन करने का पहले से निर्णय ले लेता है। इस प्रकार से PLC कठिन परिस्थिति में भी कन्ट्रोल इंजीनियर को मशीन ऑपरेशन तथा प्रोग्राम की प्रोसेस में मदद करती है।

(iv) यह त्रुटि तथा वायरस को चैक करती है। इस प्रकार से PLC मैमोरी, प्रोसेसर तथा I/O सिस्टम को एक निश्चित समय में टेस्ट करती है तथा इसके मशीन और सिस्टम को एक एक्जीक्यूशन प्रोग्राम मे शामिल करती है।

(v) PLC के द्वारा मॉनीटरिंग क्षमता को इन्डीकेशन लेम्प व इनपुट तथा आउटपुट द्वारा दर्शाया जाता है, फिर इसे एक्जीक्यूशन प्रोग्राम स्टेटस द्वारा दर्शाया जाता है।

(vi) यह टेम्प्रेचर, आर्द्रता, वाइब्रेशन तथा नॉइस को प्राप्त करती है तथा उन्हें वातावरण से प्रभावित नहीं होने देती।

(vii) इस प्रकार से PLC को किसी प्रभावी कन्ट्रोल टास्क के लिए बनाया जाता है जो किसी निश्चित कार्य में भी उपयोग की जाती है।

## P.L.C. तथा कम्प्यूटर में अन्तर (Difference between Computer and P.L.C.)

| क्र० सं० | पी०एल०सी० (PLC) | कम्प्यूटर (Computer) |
|---|---|---|
| (i) | PLC एक इलेक्ट्रॉनिक डिवाइस है जो डिजिटल सिग्नल पर ऑपरेट होती है। | कम्प्यूटर एक माइक्रोप्रोसेसर आधारित डिवाइस है जो गणितीय क्रियाओं को सम्पन्न करती है। |
| (ii) | PLC सभी क्रियाओं को ऑन लाइन स्क्रीन पर दर्शाता है। | इसमें सभी क्रियायें इसके आन्तरिक भाग में सम्पन्न होती हैं। |
| (iii) | PLC के द्वारा किसी भी ऑपरेशन को कुछ ही मिली सेकण्ड में कार्यान्वित किया जाता है। इसमें स्कैन का समय 1/60 सेकण्ड से भी कम होता है। | इसमें सम्भवत: जटिल प्रोग्राम सम्पन्न होते हैं तथा टाइमिंग ज्यादा क्रिटिकल नहीं होती है। |
| (iv) | इसको किसी विशिष्ट उद्देश्य के लिए उपयोग में लाया जाता है तथा इसमें एक ही प्रोग्राम की पुनरावृत्ति होती है। | इसमें सभी फंक्शनों के लिए अलग-अलग प्रोग्राम होते हैं। |
| (v) | इसमें पैरीफेरल डिवाइस का उपयोग नहीं किया जाता है। | इसमें विभिन्न प्रकार की पैरीफेरल डिवाइसों का उपयोग किया जाता है। |
| (vi) | इसमें सभी प्रोग्राम लैडर प्रोग्रामिंग की सहायता से लिखे जाते हैं। | इसमें सभी प्रोग्राम असेम्बली भाषा में लिखे जाते हैं। |

| | | |
|---|---|---|
| (vii) | इसके द्वारा किसी मशीन या यंत्रों को ऑटोमेटिक कन्ट्रोल किया जाता है। | इसमें किसी विशेष डिवाइस की सहायता से उससे सम्बन्धित डिवाइसों को कन्ट्रोल किया जाता है। |
| (viii) | इसमें इन्टरफेसिंग के लिए विभिन्न प्रकार के वायरों का उपयोग किया जाता है। | इसमें इन्टरफेसिंग के लिए डिवाइस का उपयोग किया जाता है। |

## P.L.C. का क्रोनोलॉजिकल इवोल्यूशन (Chronological Evolution of P.L.C.)

प्रारम्भिक समय में मशीनें मैकेनिकल माध्यमों, जैसे—केम्स, गीयर्स, लीवर्स तथा अन्य बेसिक मैकेनिकल डिवाइसों द्वारा कन्ट्रोल की जाती थी। जैसे-जैसे उत्पादन का क्षेत्र तथा मानवीय विकास बढ़ता गया उसी क्रम में कन्ट्रोल सिस्टम का भी विकास बढ़ता गया। इस प्रकार के सिस्टम एलीमेन्ट्स वायर्ड थे जो रिले तथा स्विच आदि का उपयोग करते थे। इस प्रकार के सिस्टमों का उपयोग विशेष प्रकार के मशीन ऑपरेशन तथा कन्ट्रोल लॉजिक प्रदान करने में किया जाता था। इस प्रकार के सिस्टमों का उपयोग उन प्लांटों या मशीनों में किया जाता है, जिन्हें कभी भी परिवर्तित या संशोधित नहीं करना है। परन्तु जैसे-जैसे निर्माण कार्यों की तकनीकों को बढ़ाने एवं नये प्रोडक्टो का अविष्कार हुआ, वैसे-वैसे इक्यूपमेन्ट्स को कन्ट्रोल करने के लिए और अधिक उपयोगी माध्यमों को विकसित करने की आवश्यकता होने लगी। इस प्रकार से मैकेनिकल मशीनों का कन्ट्रोल करने के लिए रिले तथा स्विचों को हार्डवायर्ड किया जाता था जो लॉजिकल संसोधन के लिए उपयुक्त नहीं थे। किसी अलग ऑपरेशन को परफॉर्म करने के लिए सिस्टमों की वायरिंग को बदलना पड़ता था जिसमें अधिक समय खर्च होता था। इस प्रकार से किसी सिस्टम के छोटी सी 'बग' को ठीक करने के लिए सिस्टम की रिवायरिंग करनी पड़ती थी, इसके बाद इसे किसी उपयोगी कार्य में उपयोग किया जाता था।

PLC का क्रमबद्ध क्रोनोलॉजिकल इवोल्यूशन निम्नलिखित प्रकार से है—

(i) सन् 1959 में एलिन ब्रेडली ने प्रथम सॉलिड स्टेट कन्ट्रोल सिस्टम के स्थान पर रिले को उपयोग में लाया जो रिप्रोग्राम्ड के योग्य नहीं था।

(ii) सन् 1968 में PLC को बैडफोर्ड ऐसोसियेट्स द्वारा डिजाइन किया गया जो जनरल मोटर्स कॉर्पोरिशन के लिए मोडीकॉन नाम से जाना जाने लगा। उन्होंने कन्ट्रोल सर्किट तथा उसकी असेंम्बली लाइन की जगह पर महँगी रिले का उपयोग किया।

(iii) सन् 1969 में प्रथम PLC मोडीकॉन द्वारा ऑटोमेटिव इन्डस्ट्रीज के लिए मैन्यूफैक्चर क्रिया, जिसे उन्होंने जनरल मोटर्स में रिले बेस्ड सिस्टम के इलेक्ट्रॉनिक सर्किट के समान उपयोग किया तथा जिसे हाई हेट 084 नाम दिया। इस प्रकार से एलिन ब्रेडली के द्वारा PLC को जनरल मोटर्स के लिए विकसित किया, लेकिन उन्होंने इसे स्वीकार नहीं किया।

(iv) सन् 1970 में मोडीकॉन तथा मॉडयूल बनाने के लिए PLC के स्पेशल फंक्शनों में अनेक परिवर्तन किये गये।

(v) सन् 1971 में प्रथम PLC का उपयोग आटोमेटिव इन्डस्ट्रीज के बाहर हुआ।

(vi) सन् 1972 में मोडीकॉन ने पूर्ण प्रोग्रामेबल PLC को लैडर लॉजिक तथा सॉलिड I/O को माइक्रोकम्प्यूटर प्रोसेसर के साथ परिचय कराया।

(vii) सन् 1973 को "स्मार्ट" PLC का परिचय दिया गया जिसे अर्थमैटिक ऑपरेशन्स, प्रिंटर कन्ट्रोल, डाटामूव तथा मैट्रिक्स ऑपरेशन आदि में उपयोग किया गया जो CRT से इन्टरफेस कर सकती थी।

(viii) सन् 1975 में एलिन ब्रेडली ने PLC-2 का परिचय कराया जो एडवांस कन्ट्रोल तथा मिनी कम्प्यूटर के साथ इन्टरफेस कर सकती थी।

(ix) सन् 1976 में पहली बार PLC का उपयोग इन्टीग्रल मैन्यूफैक्चरिंग सिस्टम में किया गया।

(x) सन् 1977 में एक बहुत छोटी माइक्रोप्रोसेसर आधारित तकनीक युक्त PLC बनायी गयी।

(xi) सन् 1978 में PLC को बड़ी स्वीकारता प्राप्त हुई।



(xii) सन् 1979 में ऐलिन ब्रेडली द्वारा PLC-2/20 का विकास किया गया जो एडवांस कन्ट्रोल सुविधाओं के साथ फॉल्ट को आसानी से ढूँढ़ने, एनालॉग मॉडयूल, सिग्नल इन्टरफेस तथा पोजीशनिंग कन्ट्रोल प्राप्त करने में सफल हुई।

(xiii) सन् 1980 में इन्टेलीजेन्ट इनपुट तथा आउटपुट मॉडयूल का विकास हुआ जिसका उपयोग हाईस्पीड तथा एक्यूरेट कन्ट्रोल में किया जाने लगा।

(xiv) सन् 1981 में एलिन ब्रेडली ने मिनी PLC-2/15 तथा मीडियम साइज PLC-2/30 का विकास किया जो मैथमैटिकल कैलकुलेशन, काउन्टिंग, टाइमर, सिक्वेन्सर, वर्डफाइल, ब्लॉक ट्रांसफर, डाटा हाइवे कनेक्टीविटी तथा ऑन लाइन एवं ऑफ लाइन प्रोग्रामिंग जैसी क्रियाओं को सम्पन्न करने में सफल हुई। इसी वर्ष उन्होंने लार्ज सिस्टम PLC को विकसित किया, जो प्रोग्राम को नॉर्मल स्कैन तथा I/O को अपडेट कर सकती थी। इस प्रकार से इस PLC के द्वारा डाटा को बाइनरी में मैनिपुलेट करना, बाइनरी डाटा को BCD में कन्वर्ट करना, इन्टीजर तथा फ्लोटिंग पॉइन्ट आदि क्रियायें सम्पन्न की जाने लगीं।

(xv) सन् 1982 में लार्जर PLC को 8192 I/O के साथ विकसित किया गया।

(xvi) सन् 1983 में PLC को "थर्ड पार्टी" पैरीफेरल्स, ग्राफिक्स CRT, ऑपरेटर इन्टरफेस, स्मार्ट I/O नेटवर्क, पैनल डिस्प्ले तथा डॉक्यूमेन्टेशन आदि के साथ विकसित किया गया।

(xvii) सन् 1984 में एलिन ब्रेडली ने PLC-4 को विकसित किया जो एक छोटी सी यूनिट के द्वारा 20 इनपुट तथा 12 आउटपुट देने में सक्षम थी। इस प्रकार से छोटी-छोटी PLCs को आपस में इन्टरकनेक्ट करके पूरे प्लांट को कन्ट्रोल किया जाने लगा।

(xviii) सन् 1990 के दशक में PLC का उपयोग सभी प्लांटों में किया जाने लगा। इसमें विभिन्न प्रकार की I/O डिस्ट्रीब्यूशन डिवाइसों, I/O पैरीफेरल्स को सेन्ट्रल में कनेक्ट कर कन्ट्रोलिंग की जाने लगी तथा 2 से 4 वायर की केबिल्स की सहायता से कम्युनिकेशन प्रोसेस को बढ़ाया गया।

(xix) सन् 2000 से आज तक विभिन्न प्रकार की डिजिटल कम्युनिकेशन डिवाइसों के माध्यम से PLC को इन्टरफेस कराया जाने लगा। इसकी सहायता से ऑन लाइन प्रोडक्शन प्रोसेसिंग तथा अन्य प्रकार की प्रक्रियाओं को सम्पन्न किया जाने लगा। अतः आजकल डिजिटल कम्यूनिकेशन तथा प्रोग्रामेबल लॉजिक कन्ट्रोलर की सहायता से प्रत्येक इलेक्ट्रॉनिक डिवाइस को आसानी से ऑपरेट तथा कन्ट्रोल किया जा सकता है।

## P.L.C. की हानियाँ (Disadvantages of P.L.C.)

**(i) नई तकनीक** (New Technology)—नई तकनीक आने पर PLC के रिले तथा लैडर को बदलना कठिन होता है। इस प्रकार से नई तकनीक आने पर सम्पूर्ण सिस्टम तथा सॉफ्टवेयर में परिवर्तन करना पड़ता है जिसे इलेक्ट्रीशियन तथा टेक्नीशियन को नई तकनीक से सम्बन्धित कोर्स को सीखना पड़ता है।

**(ii) वातावरण पर सोच-विचार** (Environmental Considerations)—PLC के उपयोग से उच्च ताप उत्पन्न होता है जो अन्य इलेक्ट्रॉनिक डिवाइसों तथा इन्डस्ट्रियल वातावरण को हानि पहुँचाता है।

**(iii) बेरोजगारी** (Unemployment)—PLC के उपयोग से किसी प्रोसेस को बहुत ही जल्दी तथा कम लोगों के द्वारा सम्पन्न किया जाता है जिससे बेरोजगारी बढ़ती है।

**(iv) फेल-सेफ ऑपरेशन** (Fail-Safe Operation)—रिले ऑपरेशन में, स्टॉप बटन सर्किट को इलेक्ट्रीकली डिसकनेक्ट करता है। यदि पावर फेल हो जाती है तो सिस्टम रुक जाता है। इसके अतिरिक्त रिले सिस्टम में पावर के आ जाने पर यह पुनः रीस्टार्ट नहीं होता है, इसके लिए एक निश्चित PLC प्रोग्राम का उपयोग किया जाता है।

**(v) मॉडयूल्स की आवश्यकता** (Requirement of Modules)—PLC को विभिन्न इनपुट एवं आउटपुट डिवाइसों से इन्टरफेस करने के लिए मॉडयूल्स की आवश्यकता होती है जो आसानी से उपलब्ध नहीं होते हैं।

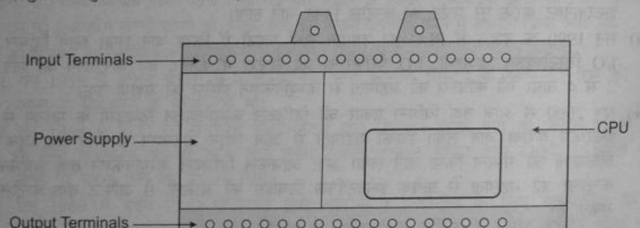
**(vi) डिजाइन की परेशानी** (Difficulty of Design)—PLC को रिले लैडर लॉजिक के लिए निर्मित किया जाता है। इसको कुछ एडवांस डिवाइसों के साथ इन्टरफेस करने के लिए सिग्नल प्राप्त नहीं होते हैं।

**(vii) ट्रेनिंग की परेशानी** (Difficulty of Training)—प्रोग्रामिंग, ऑपरेशन एवं PLC की ट्रबलशूटिंग के लिए ट्रेनिंग की आवश्यकता होती है जो आसानी से उपलब्ध नहीं है। इसके लिए किसी अच्छे इन्स्टीट्यूट या मैन्यूफैक्चरर से सम्पर्क करना पड़ता है।

## P.L.C. के प्रकार (Types of P.L.C.)

संरचना के आधार पर PLC को निम्नलिखित दो भागों में बाँटा जा सकता है—

**(i) फिक्स्ड इनपुट/आउटपुट PLC** (Fixed Input/Output PLC)—एक फिक्स्ड PLC, फिक्सड अथवा बिल्ट-इन इनपुट तथा आउटपुट सेक्शन्स से मिलकर बनी होती है। इस प्रकार के फिक्स्ड PLC के इनपुट/आउटपुट तथा बिल्ट-इन अपरिवर्तनीय होते हैं। इसमें एक फिक्स्ड, बिल्ट-इन तथा नॉन रिमूवेबल स्क्रू टर्मिनल स्ट्रिप होती है जो सभी प्रकार के इनपुट सिग्नल के स्क्रू टर्मिनल्स कनेक्शन तथा सभी आउटपुट समाहित किये होती है। इसमें लेवल्ड क्षेत्र पर "इनपुट टर्मिनलों" का एक प्लास्टिक गेट होता है जो ऊपर की ओर स्थित रहता है। इस गेट को खोलकर सभी बिल्ट-इन इनपुट स्क्रू टर्मिनल्स को प्रदर्शित किया जाता है। "आउटपुट टर्मिनल्स" से लेवल्ड प्लास्टिक गेट नीचे की ओर स्थित होता है। इस गेट को खोलकर सभी बिल्ट-इन आउटपुट स्क्रू टर्मिनल्स को प्रदर्शित किया जाता है। चित्र 1.2 में फिक्सड इनपुट/आउटपुट PLC को प्रदर्शित किया गया है।



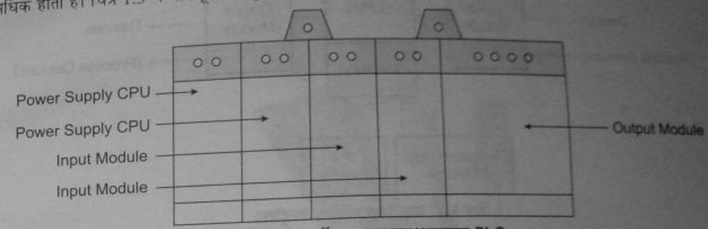
चित्र 1.2 : फिक्स्ड इनपुट/आउटपुट PLC

इसमें नेनो एवं माइक्रो दो फिक्सड PLC होती हैं। नेनो PLC छोटा आकार की PLC होती है जिसकी मैमोरी क्षमता RAM से 16 k byte कम होती है। इसमें 16 इनपुट/आउटपुट होते हैं। माइक्रो PLC फिक्स्ड PLCs होती है जिसकी मैमोरी क्षमता कुछ kB तक होती है, जिसमें 32 I/O होते हैं। इस प्रकार की PLC पावर सप्लाई, प्रोसेसर, I/O मॉडयूल्स तथा मैमोरी स्वयं में समाहित किये होती है। इनको पैकेज्ड कन्ट्रोलर्स के नाम से भी जाना जाता है। इस आकार में छोटे तथा कम मूल्य के होते हैं इसलिय इनको आसानी से इन्स्टॉल एवं खरीदा जा सकता है।

**(ii) मॉडयूलर इनपुट/आउटपुट PLC** (Modular I/O PLC)—मॉडयूलर PLC के प्रोसेसर यूनिट में टर्मिनल स्ट्रिप बिल्ट नहीं होती है। मॉडयूलर PLC में इनपुट आउटपुट पॉइन्ट्स को प्लगइन टाइप रिमूवेबल कहा जाता है। मॉडयूलर PLC के इनपुट एवं आउटपुट के साथ चेसिस, रैक अथवा बेसप्लेट शामिल किये जाते हैं, जहाँ पर पावर सप्लाई, CPU तथा सभी इनपुट आउटपुट मॉडयूल्स के हार्डवेयर आइटम्स को अलग करने के लिए उपस्थित रहती है। प्रत्येक मॉडयूल तथा CPU के बीच इलेक्ट्रिकल कनेक्शन दो मेटिंग प्लग द्वारा बनाया जाता है। इस प्रकार से इस प्लग को PLC मॉडयूल से जोड़ा जाता है जिसका उपयोग मॉडयूल से CPU को सिग्नल ट्रांसफर करने के लिए किया जाता है। अन्य को बेसप्लेट पर क्लिपिंग करने के लिए उपयोग में लाया जाता है जिससे कि यह रैक तथा चेसिस पर स्लाइड न कर सकें।



पावर सप्लाई, मॉडयूलर, लाइन वोल्टेज तथा रैक का चेसिस पर दायीं या बायीं ओर या अन्य किसी पोजीशन पर हुकड-अप होती है। CPU को पावर सप्लाई के बगल में दायीं या बायीं पोजीशन में इन्स्टॉल किया जाता है। यह विशेष रूप से एक प्लग इन मॉडयूल है। यह मध्यम आकार एवं बड़े PLC मॉडयूलर होते हैं। मध्यम आकार की PLC के इनपुट्स/आउटपुट्स की गणना 1024 kB से भी कम होती है तथा बड़े आकार की PLCs की गणना 1024 kB से अधिक होती है। चित्र 1.3 में मॉडयूलर इनपुट/आउटपुट PLC दर्शायी गई है।



चित्र 1.3 : मॉडयूलर इनपुट/आउटपुट PLC

मॉडयूलर PLC को निम्नलिखित भागों में बाँटा जा सकता है—

**( अ ) स्मॉल PLC** (Small PLC)—स्मॉल PLC को डिजाइन करने पर 100 से कम इनपुट तथा आउटपुट पोर्ट बनाये जाते हैं। इसमें 20 इनपुट तथा 12 आउटपुट पोर्ट प्रोसेसर के साथ लोकली जोड़े जाते हैं। इस प्रकार के डिजाइन में अतिरिक्त इनुपुट/आउटपुट रिमोट की सहायता से रैक्स से एक्मोडेट पर जोड़े जाते हैं। इस प्रकार की PLC 2kB से 10 kB लॉजिकल प्रोग्राम स्टोर कर सकती है।

**( ब ) मीडियम PLC** (Medium PLC)—इस प्रकार की PLC में इन्स्ट्रक्शन सेट, जैसे—मैथमैटिकल फंक्शन, फाइल फंक्शन्स तथा PID प्रोसेस कन्ट्रोल आदि इन्क्लूड किये जाते हैं। इस प्रकार की PLCs में 4000 से 8000 तक इनपुट्स तथा आउटपुट होते हैं। ये सभी स्पेशल मॉडयूल के तत्वों, जैसे—ASCII कम्युनिकेशन मॉडयूल्स, BASIC प्रोग्रामिंग मॉडयूल्स, 16-बिट मल्टीप्लेक्सिंग मॉडयूल्स, एनालॉग इनपुट तथा आउटपुट मॉडयूल्स एवं कम्यूनिकेशन मॉडयूल्स आदि के लिए बनाये जाते हैं।

**( स ) लार्ज PLC** (Large PLC)—लार्ज PLCs को संपूर्ण इन्डस्ट्रीज ऑटोमेशन को पर्याप्त मैमोरी स्पेस देने के लिए बनाया गया है। यद्यपि इसकी मुख्य हानि यह है कि जब पूर्ण इन्डस्ट्रीज बंद हो जाती है तब PLC असंक्रिया सम्पन्न करती है। इस प्रकार के सिस्टम में लोकर एरिया नेटवर्क को डिस्ट्रीब्यूटिड कन्ट्रोल में उपयोग किया जाता है तथा छोटी PLC को आपस में जोड़कर एक निश्चित नेटवर्क को उत्पन्न किया जाता है। इस प्रकार से किसी इन्डस्ट्रीज में विभिन्न PLCs उपयोग में लायी जाती हैं तथा सम्पूर्ण सिस्टम के फेल हो जाने से उसके एक निश्चित भाग को PLC द्वारा प्रभावित नहीं होने दिया जाता है।

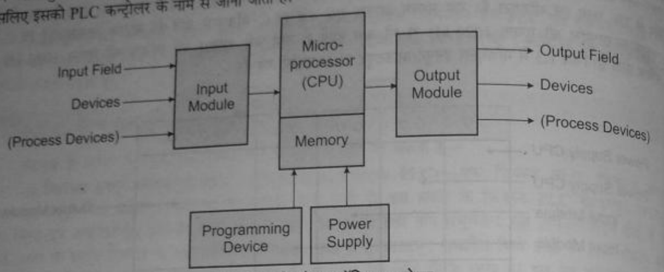
## P.L.C. का ब्लॉक डायग्राम (Block Diagram of P.L.C.)

प्रोग्रामेबल लॉजिक कन्ट्रोलर एक स्पेशल कम्प्यूटर की तरह होता है जिसमें सभी प्रकार के बेसिक फंक्शन कम्प्यूटर के समान ही होते हैं। प्रोग्रामेबल लॉजिक कन्ट्रोलर का उपयोग मुख्य रूप से इन्डस्ट्रियल ऑपरेशनों में किया जाता है। प्रोग्रामेबल लॉजिक कन्ट्रोलर को ब्लॉक डायग्राम 1.3 में दर्शाया गया है।

प्रोग्रामेबल लॉजिक कन्ट्रोलर के प्रमुख भाग निम्नलिखित हैं—

**(i) प्रोसेसर** (Processor)—सभी प्रोग्रामेबल लॉजिक कन्ट्रोलर एक सेन्ट्रल प्रोसेसिंग यूनिट (CPU) को रखते हैं यह माइक्रोप्रोसेसर आन्तरिक क्रिया को सुपरवाइज तथा कन्ट्रोल करता है। इस प्रकार से CPU यूजर्स के प्रोग्राम्स को ह

एवं आउटपुट के स्टोरेज को अपडेट करता है। यह विभिन्न लॉजिक एवं सिक्वेसिंग फंक्शन को कार्यान्वित करता है इसलिए इसको PLC कन्ट्रोलर के नाम से जाना जाता है।




चित्र 1.4 : प्रोग्रामेबल लॉजिक कन्ट्रोलर

प्रोग्राम एक्जीक्यूशन के दौरान प्रोसेसर इनपुट्स को रीड करता है, उनकी वेल्यूस लेता है तथा कन्ट्रोल एप्लीकेशन प्रोग्राम के अनुसार आउटपुट को सक्रिया अथवा निष्क्रिय करता है। इस प्रकार से लैडर नेटवर्क को हल किया जाता है। इसमें लॉजिक एक बार हल हो जाने के बाद प्रोसेसर आउटपुट को अपडेट करता है। इसमें इनपुट को पढ़ने, कन्ट्रोल एप्लीकेशन प्रोग्राम को कार्यान्वित करने एवं आउटपुट को अपडेट करने की प्रक्रिया स्कैन के नाम से जानी जाती है। PLCs को स्कैन टाइम से वर्गीकृत किया जाता है। सामान्यत: स्कैन टाइम की सीमा 1 से 100 मिली सेकण्ड तक होती है। इसमें Z-80, 8086, 9900, 6800, 286, 386, 486 माइक्रोप्रोसेसर फैमिली के कुछ CPUs उपयोग किये जाते हैं।

**(ii) मैमोरी** (Memory)—मैमोरी के अन्तर्गत लॉजिक प्रोगाम्स, सिक्वेसिंग एवं अन्य इनपुट/आउटपुट ऑपरेशन सम्मिलित किये जाते हैं। PLC मैमोरी की स्टोर करने की क्षमता 1 kB से 64 kB तक होती है। PLC को सिस्टम मैमोरी (ऑपरेटिंग सिस्टम) तथा एप्लीकेशन मैमोरी में विभाजित किया जा सकता है। इस प्रकार से सिस्टम प्रोग्राम्स को ROM तथा एप्लीकेशन प्रोग्राम को RAM में स्टोर किया जाता है।

**(iii) प्रोग्रामिंग डिवाइस** (Programming Device)—PLC को एक प्रोग्रामिंग डिवाइस के माध्यम से प्रोग्राम्ड किया जाता है। प्रोग्रामिंग डिवाइस के द्वारा प्रोग्रामर अथवा ऑपरेटर प्रोग्राम इन्स्ट्रक्शन्स तथा डाटा को एन्टर व एडिट कर सकते हैं। प्रोग्रामिंग डिवाइस के बेसिक अवयव कीबोर्ड, विजुअल डिस्प्ले, माइक्रोप्रोसेसर तथा कम्युनिकेशन केबिल होती हैं। PLC को केबिनेट से अलग किा जाता है जिससे कि इसके भागों को अलग-अलग कन्ट्रोलर्स के मध्य बांटा जा सके। विभिन्न PLC निर्माता विभिन्न प्रकार की डिवाइसेज उपलब्ध करते हैं। यह डिवाइसेज हैण्डहेल्ड प्रोग्रामिंग यूनिट या पर्सनल कम्प्यूटर या इन्डस्ट्रियल प्रोग्रामिंग टर्मिनल हो सकती हैं। सामान्यत: प्रोग्रामिंग डिवाइस प्रोग्रामिंग के समय कन्ट्रोल सिस्टम की ट्रबलशूटिंग से कनेक्ट रहती है, अन्यथा यह प्रोग्रामिंग डिवाइस से डिस्कनेक्ट हो जाती है। प्रोग्रामिंग टर्मिनल के कुछ महत्वपूर्ण फंक्शन्स निम्नलिखित हैं—

(अ) PLCs में मैमोरी से यूजर प्रोग्राम को उत्पन्न एवं ट्रांसफर करना,
(ब) यूजर प्रोग्राम तथा कन्ट्रोल सिस्टम के स्टार्टअप को डीबग करना,
(स) इन्स्टॉलेशन डायग्नोस्टिक्स को परफॉर्म करना।

प्रोग्रामिंग डिवाइस में उपयोग होने वाली प्रमुख डिवाइसेज निम्नलिखित हैं—

**(अ) हैण्डहेल्ड प्रोग्रामिंग यूनिट** (Handheld Programming Unit)—हैण्डहेल्ड प्रोग्रामर्स इनएक्सपेन्सिव एवं पोर्टेबल यूनिट्स होती हैं जिनको सामान्य रूप से छोटी PLCs के प्रोग्राम हेतु उपयोग में लाया जाता है। इसमें से अधिकतर यूनिट्स एक बड़े डिस्प्ले एवं एक अलग कीबोर्ड तथा कैलकुलेटर के साथ पोर्टेबल होती हैं। सामान्यत: इनकी



डिस्प्ले LED अथवा LCD मैट्रिक्स होती है तथा इस प्रकार की डिवाइस के कीबोर्ड में अल्फान्यूमेरिक कीज, प्रोग्रामिंग इन्स्ट्रक्शन्स कीज तथा स्पेशल फंक्शन कीज आदि होती हैं। इसके अतिरिक्त यह मुख्य रूप से कन्ट्रोल प्रोग्राम की इनपुटिंग एवं एडीटिंग के लिए उपयोग की जाती है। प्रोग्रामेबल कन्ट्रोल भी टेस्टिंग, चैकिंग एवं प्रोग्राम की मॉनीटरिंग के लिए उपयोग किये जाते हैं। हैण्डहेल्ड प्रोग्रामिंग यूनिट को चित्र 1.5 में दर्शाया गया है।

चित्र 1.5 : हैण्डहेल्ड प्रोग्रामिंग यूनिट

**(ब) इन्डस्ट्रियल प्रोग्रामिंग टर्मिनल** (Industrial Programming Terminal)—इन्डस्ट्रियल प्रोग्रामिंग टर्मिनल एक इन्टेलीजेंस डिवाइस है जो कन्ट्रोल प्रोग्राम को डिस्प्ले करने के साथ-साथ प्रोग्राम एडीटिंग फंक्शन्स भी उपलब्ध कराती है। यह फंक्शनल प्रोग्रामेबल कन्ट्रोलर्स से स्वतन्त्र होती है। इन्डस्ट्रियल प्रोग्राम में सामान्यत: एक स्वयं की आन्तरिक मैमोरी तथा कैथोड रे ट्यूब (CRT) होती है जो एल्टर एवं मॉनीटर प्रोग्राम्स को उत्पन्न करने में उपयोग की जाती है।

CRT प्रोग्रामिंग के लिए एक शक्तिशाली टूल है क्योंकि कन्ट्रोल प्रोग्राम्स को प्रोग्रामेबल कन्ट्रोलर से जोड़े बिना संशोधित एवं प्रदर्शित नहीं किया जा सकता है। प्रोग्रामिंग टर्मिनल को चित्र 1.6 में प्रदर्शित किया गया है।

चित्र 1.6 : इन्डस्ट्रियल प्रोग्रामिंग टर्मिनल

**(स) पर्सनल कम्प्यूटर** (Personal Computer)—प्रोग्रामिंग टर्मिनल्स में आधुनिक परिवर्तन को पर्सनल कम्प्यूटर टाइप कहा जाता है। पर्सनल कम्प्यूटर के अन्दर इन्डस्ट्रियल टर्मिनल्स के सभी फीचर्स, जैसे—प्रोग्राम एडिटिंग एवं स्टोरेज आदि आते हैं। परन्तु इनके द्वारा कुछ अन्य फीचर्स, जैसे—ऑटोमेटिक प्रोग्राम, प्रिन्टआउट एवं लोकल एरिया नेटवर्क आदि को जोड़ा जाता है। LAN, प्रोग्रामर अथवा इंजीनियर को नेटवर्क में किसी भी प्रोग्रामेबल कन्ट्रोलर्स के एक्सेस करने की अनुमति प्रदान करता है ताकि नेटवर्क से किसी भी डिवाइस को मॉनीटर अथवा कन्ट्रोल किया जा सके। इस प्रकार के फंक्शन में छोटे पोर्टेबल कम्प्यूटर, लैपटॉप आदि PLC की प्रोग्रामिंग में उपयोग किये जाते हैं। पर्सनल कम्प्यूटर को चित्र 1.7 में दर्शाया गया है।

चित्र 1.7 : पर्सनल कम्प्यूटर

**(iv) इनपुट मॉडयूल** (Input Module)—इनपुट मॉडयूल फील्ड इनपुट डिवाइसेज तथा PLCs के CPU के मध्य लिंक की तरह कार्य करता है। प्रत्येक मॉड्यूल की इनपुट वायरिंग को प्रत्येक अलग-अलग फील्ड इनपुट डिवाइस से जोड़ने के लिए एक टर्मिनल ब्लॉक होता है। इनपुट मॉड्यूल का मुख्य फंक्शन यह है कि यह फील्ड डिवाइस के इनपुट सिग्नल को लेकर एक ऐसे सिग्नल लेवल में परिवर्तित करता है जिससे CPU उससे सम्बन्धित कार्य कर सके एवं इस सिग्नल को इलेक्ट्रिकली अलग करके बैकप्लेन के रास्ते CPU को भेज सके।

विशेष रूप से इनपुट मॉडयूल में 8, 16 या 32 इनपुट टर्मिनल्स होते हैं। इनपुट मॉड्यूल एक इलेक्ट्रॉनिक सर्किट होता है जो PLC को CPU की फील्ड इनपुट डिवाइसेज के साथ इन्टरफेस कराता है।

**(v) आउटपुट मॉडयूल** (Output Module)—आउटपुट मॉडयूल PLC के CPU तथा फील्ड आउटपुट डिवाइसेज के मध्य एक लिंक की तरह कार्य करता है। प्रत्येक मॉडयूल के पास आउटपुट वायरिंग को अलग-अलग फील्ड आउटपुट डिवाइसेज से जोड़ने के लिए एक टर्मिनल ब्लॉक होता है। आउटपुट मॉड्यूल का मुख्य फंक्शन CPU के कन्ट्रोल सिग्नल को ग्रहण एवं इलेक्ट्रिकली अलग-अलग करना तथा आउटपुट फील्ड डिवाइस को ऑन अथवा ऑफ करने के लिए सक्रिय अथवा निष्क्रिय करना है। विशेष रूप से आउटपुट मॉड्यूल में 8, 16 या 32 आउटपुट टर्मिनल्स होते हैं। आउटपुट मॉड्यूल एक इलेक्ट्रॉनिक सर्किट होता है जो PLC की सेन्ट्रल प्रोसेसिंग यूनिट की फील्ड आउटपुट डिवाइसेज के साथ जुड़ा रहता है।

**(vi) पावर सप्लाई** (Power Supply)—पावर सप्लाई, प्रोग्रामेबल कन्ट्रोलर सिस्टम में इलेक्ट्रॉनिक सर्किट्स को पावर देने के लिए AC लाइन वोल्टेज को DC लाइन वोल्टेज में परिवर्तित करती है। इस पावर सप्लाई में सिस्टम को वोल्टेज एवं करंट की सही मात्रा सप्लाई करने के लिए रेक्टीफायर, फिल्टर, रेगुलेटर वोल्टेज करंट को सम्मिलित करता है। पावर सप्लाई द्वारा 280 V AC लाइन वोल्टेज को DC लाइन वोल्टेज में परिवर्तित किया जाता है। PLC के लिए पावर सप्लाई को प्रोसेसर, मैमोरी एवं इनपुट/आउटपुट मॉड्यूल्स में सिंगल हाउसिंग के अन्तर्गत जोड़ा जाता है तथा यह केबिल के द्वारा जुड़ी हुई एक अलग यूनिट भी हो सकती है।

पावर सप्लाई को उच्चताप एवं नमी के लिए निर्मित किया जाता है क्योंकि PLC का उपयोग इन्डस्ट्रियल वातावरण में किया जाता है। इसका उपयोग AC पावर अथवा इन्डस्ट्रियल प्लांट्स की सिग्नल लाइन्स में उपस्थित इलेक्ट्रिकल नॉइस (noise) को हटाने के लिए किया जाता है जिससे कि नॉइस द्वारा कन्ट्रोल सिस्टम में अशुद्धि उत्पन्न न हो सके।



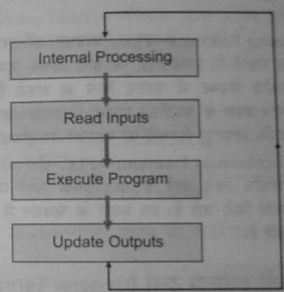
## P.L.C. की सामान्य कार्यविधि (Working of P.L.C.)

**(i) इन्टर्नल प्रोसेसिंग** (Internal Processing)—इन्टर्नल प्रोसेसिंग के अन्तर्गत मॉनीटर सिस्टम, PLC की प्रोग्राम मैमोरी, एक्जीक्यूशन क्षमता का परीक्षण तथा उसके समय को व्यवस्थित किया जाता है। इसके द्वारा वर्तमान शैड्यूल ब्लॉक तथा LED, RUN, COM, EPR को अपडेट किया जाता है। इसके द्वारा मुख्य रूप से प्रोग्रामिंग पोर्ट के प्रोसेस की रिक्वेस्ट सम्पन्न की जाती है।

**(ii) रीड इनपुट्स** (Read Inputs)—फिजिकल इनपुट्स के स्टेटस के साथ इनपुट मैमोरी को अपडेट किया जाता है।

**(iii) एक्जीक्यूट प्रोग्राम** (Execute Program)—इस प्रक्रिया में यूजर द्वारा प्रोग्राम लिखा जाता है तथा उसे एक्जीक्यूट कराया जाता है।

**(iv) अपडेट आउटपुट्स** (Update Outputs)—इसमें आउटपुट मैमोरी से फिजिकल आउटपुट के स्टेटस को अपडेट किया जाता है।




इस प्रकार के ऑपरेशन में लैडर लॉजिक द्वारा परिभाषित किये गये प्रोग्राम को चरण PLC द्वारा एक के बाद एक लगातार एक्जीक्यूट किये जाते हैं। आउटपुट्स में किसी भी प्रकार के परिवर्तन को एक्जीक्यूट कराने तथा प्रोग्राम के अनुसार कार्यान्वित करने के लिए PLC प्रोसेसर को एक निश्चित समय की आवश्यकता होती है।

इस प्रकार के फंक्शन में PLC सर्वप्रथम इन्टर्नल प्रोसेसिंग ऑपरेशन्स परफॉर्म करती है तथा इसके बाद प्रोसेसर द्वारा PLC को इनपुट सेम्पल दिये जाते हैं। इस प्रकार से उन कन्टेन्ट को मैमोरी में स्टोर किया जाता है। इसके उपरान्त कन्ट्रोल प्रोग्राम एक्जीक्यूट होता है। मैमोरी में स्टोर इनपुट वेल्यूस को कन्ट्रोल लॉजिक कैलकुलेशन्स के रूप में आउटपुट में उपयोग किया जाता है। अन्त में कैलकुलेट की गई वेल्यूस के साथ आउटपुट को एग्री (agree) किया जाता है तथा उसे अपडेट किया जाता है। इस प्रोसेस में इनपुट को पढ़ना, कन्ट्रोल प्रोग्राम को एक्जीक्यूट करना एवं आउटपुट्स को रिवाइस करना आदि चरण सम्मिलित किये जाते हैं। आउटपुट को रिवाइस करने की प्रक्रिया स्कैन कहलाती है एवं स्कैन को परफॉर्म करने में लगा समय स्कैन टाइम कहलाता है। यह समय प्रत्येक चक्र को परफॉर्म होने के लिए कन्ट्रोल फंक्शन्स की कॉम्प्लेक्सिटी एवं उसकी संख्या पर निर्भर करता है। इस प्रकार यह स्कैन टाइम लैडर डायग्राम की संख्या तथा प्रत्येक रंग परहोने वाले ऑपरेशन्स की कॉम्प्लेक्सिटी पर निर्भर करता है। सामान्यत: स्कैन टाइम 1 से 100 ms के मध्य परिवर्तित हो सकता है। PLC के लिए स्कैन टाइम फिक्स्ड होता है।

## P.L.C. के फंक्शन्स (Functions of P.L.C.)

**(i) कन्ट्रोल रिले** (Control Relay)—इलेक्ट्रिकल पावर डिवाइसेज, जैसे—हीटर, पम्प, ब्लोअर आदि के स्टार्ट अथवा स्टॉप ऑपरेशन के लिए रिलेज का उपयोग बड़ी इन्डस्ट्रीज में किया जाता है। रिलेज को PLC द्वारा सक्रिय तथा निष्क्रिय किया जाता है।

**(ii) टाइमिंग फंक्शन्स** (Timing Functions)—यह PLC टाइमर की तरह प्रदर्शन करता है तथा टाइम डिले उपलब्ध कराता है। PLC में प्रोग्रामेबल टाइमर्स उपयोग किये जाते हैं जिन्हें यूजर्स टाइमर पैरामीटर के द्वारा कम समय में आसानी से सेट किया जाता है। एक विशिष्ट समय अंतराल के बाद PLC टाइमर निश्चित प्रक्रियाओं की डिवाइसेज को ऑपरेट करते हैं तथा टाइमर ऑपरेशन्स को परफॉर्म करते हैं।

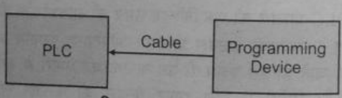
**(iii) एनालॉग कन्ट्रोल** (Analog Control)—PLC, एनालॉग सिग्नल को प्राप्त तथा उत्पन्न कर सकता है। यह फंक्शन विभिन्न एनालॉग डिवाइसेज, जैसे—तापमान, दाब, फ्लो, लेवल ट्रांसड्यूसर्स आदि के द्वारा सिग्नल प्राप्त करता है। इसकी सहातया से एनालॉग आउटपुट डिवाइसेज, जैसे—कन्ट्रोल वॉल्व, शन्ट मोटर आदि को PLC द्वारा कन्ट्रोल किया जाता है।

**(iv) काउन्टिग फंक्शन** (Counting Function)—PLC, काउन्टर की तरह एक्ट करती है तथा इन्डस्ट्रीज के फिनिश्ड प्रोडक्ट्स को भी काउन्ट कर सकती है। इसमें PLC के द्वारा बोतल, ड्रम्स बैग्स तथा व्हीकल्स आदि काउन्ट किये जा सकते हैं। इसके द्वारा डिफेक्टेड प्रोडक्ट भी काउन्ट किये जा सकते हैं। PLC में प्रोग्रामेबल काउन्टर का उपयोग किया जाता है जिसे यूजर विभिन्न प्रकार के काउन्टिंग प्रोग्रामों में उपयोग करता है। इस प्रकार से आब्जेक्ट्स को सेन्सर द्वारा सेन्स्ड किया जाता है तथा उसके आउटपुट सिग्नलों को काउन्टर पर ट्रांसफर किया जाता है।

**(v) मिसलेनियस फंक्शन** (Miscellaneous Function)—PLC द्वारा प्रोसेस डाटा पर AND, OR, NOT, XOR आदि लॉजिकल ऑपरेशन परफॉर्म किये जाते हैं तथा इसके साथ-साथ एडीशन, सबट्रेक्शन, डिवीजन, मल्टीप्लिकेशन आदि ऑपरेशन भी परफॉर्म किये जाते हैं। इस प्रकार के फंक्शन में कम्पेरीजन, सिक्वेंसर, डाटा ट्रांसफर ऑपरेशन भी PLC द्वारा परफॉर्म किये जाते हैं।

## P.L.C. में उपयोग होने वाली सामान्य टर्म्स (General Terms of P.L.C.)

**(i) डाउनलोडिंग** (Downloading)—प्रोग्रामिंग डिवाइस से PLC मैमोरी में PLC प्रोग्राम को ट्रांसफर करना डाउनलोडिंग कहलाता है।




चित्र 1.8 : डाउनलोडिंग

**(ii) रन मोड** (Run Mode)—यदि सभी इनपुट तथा आउटपुट सिग्नल्स को सही स्क्रू टर्मिनल्स के साथ वार्यड किया जाए तब प्रोसेसर को रन मोड में कहा जाता है। रन मोड में प्रोग्राम लगातार चलता रहता है एवं इसमें इन्स्ट्रक्शन्स प्रोग्राम्ड को हल किया जाता है।

**(iii) DIN रेल** (DIN Rail)—यह इलेक्ट्रिकल पैनल से जुड़ी हुई मैटल ट्रेक अथवा रेल होती है जहाँ डिवाइसेज को रेल द्वारा आसानी से क्लिक्ड अथवा रिमूव्ड किया जा सकता है। इसका विशिष्ट लाभ यह है कि इसे इन्स्टॉलेशन, मेन्टीनेन्स अथवा रिप्लेसमेन्ट के लिए आसानी से हार्डवायर्ड से अलग किया जा सकता है।

**(iv) चैसिस** (Chassis)—चैसिस पावर सप्लाई, प्रोसेसर एवं इनपुट/आउटपुट मॉड्यूल्स को एक यूनिट की तरह साथ-साथ रखता है। चैसिस में बैकप्लेन सम्मिलित रहता है जो इनपुट/आउटपुट मॉड्यूल्स एवं प्रोसेसर के मध्य कन्ट्रोल सिग्नल्स ट्रॉंसफर करता है। निर्माणकर्त्ता इसको चैसिस, टैक अथवा बेसप्लेट कहते हैं।



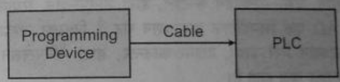
**(v) प्रोसेसर स्कैन** (Processor Scan)—यह प्रोसेसर में चलता हुआ एक क्रम है जो निम्नलिखित कार्य करता है—

(अ) सभी इन्स्ट्रक्शनों का मूल्यांकन करना।
(ब) इनपुट स्टेटस टेबल में इनपुट कन्डीशन्स को स्टोर करना।
(स) यूजर प्रोग्राम को हल करना।
(द) आउटपुट स्टेटस फाइल को अपडेट करना।
(इ) अन्य PLC के मध्य कम्यूनिकेशन को अपडेट करना।
(फ) हाउस कीपिंग कार्यों को परफॉर्म करना तथा वॉचडॉग टाइमर को रीसेट करना।

**(vi) प्रोग्राम स्कैन** (Program Scan)—प्रोग्राम स्कैन, PLC प्रोसेसर स्कैन ऑपरेटिंग साइकिल का एक भाग है। प्रोसेसर स्कैन के दौरान, CPU यूजर प्रोग्राम के प्रत्येक रंग को स्कैन करता है। इस प्रक्रिया को प्रोग्राम स्कैन कहा जाता है।

**(vii) वॉचडॉग टाइमर** (Watchdog Timer)—वॉचडॉग टाइमर में एक हार्डवेयर टाइमर का उपयोग किया जाता है जो यह निश्चित करता है कि प्रोग्राम सही टाइम पर स्कैन हुआ है या नहीं।

**(viii) अपलोडिंग** (Uploading)—PLCs मैमोरी के यूजर प्रोग्राम्स को कम्प्यूटर से अन्य प्रोग्रामिंग डिवाइसों में ट्रांसफर करना अपलोडिंग कहलाता है।




चित्र 1.9 : अपलोडिंग

**(ix) प्रोग्राम मोड** (Program Mode)—यूजर प्रोग्राम को डाउनलोड करने से पूर्व प्रोसेसर को प्रोग्राम मोड में होना चाहिए।

**(x) बौड रेट** (Baud Rate)—इसके द्वारा बाइनरी डाटा की बिट प्रति सेकण्ड संख्या परिभाषित की जाती है, जिनको PLC एवं अन्य पैरीफेरल्स के मध्य सीरियल कम्युनिकेशन के दौरान प्राप्त एवं स्थानान्तरित किया जाता है। यह रेट 50 से 19200 बौड तक परिवर्तित हो सकती है। इसकी सबसे अधिक सामान्य रेट 300, 1200 एवं 9800 बौड होती है।

## P.L.C. की मेन्टीनेन्स (Maintenance of P.L.C.)

(i) इनपुट/आउटपुट मॉड्यूल टर्मिनल की टाइटनेस समय-समय पर चेक करनी चाहिए क्योंकि ये कुछ समय पश्चात् लूज हो जाती हैं।
(ii) PLC का मेन्टीनेन्स करते समय पावर सप्लाई को ऑफर रखना चाहिए।
(iii) PLC के टर्मिनल का समय-समय पर कोरोजन (Corrosion) चैक करना चाहिए। क्योंकि नमी तथा कोरोसिव वातावरण के कारण इलेक्ट्रिकल कनेक्शन को कमजोर कर देता है।
(iv) सभी कम्पोनेन्ट को धूल से मुक्त रखना चाहिए तथा उन सभी में सही तरीके से कूलिंग देनी चाहिए जिससे कि धूल अन्दर न जा सके।
(v) विभिन्न स्पेयर पार्टसों को स्टॉक में स्टोर करके रखना चाहिए ताकि इनपुट तथा आउटपुट मॉड्यूल के फेल हो जाने पर इन्हें उपयोग में लाया जा सके।
(vi) ऑपरेटिंग प्रोग्राम का डुप्लीकेट रिकॉर्ड पेपर, टेप या कम्प्यूटर डिस्क पर रखना चाहिए जिससे कि उसे मेन्टीनेन्स प्रोसेस के समय उपयोग मे लाया जा सके।

(vii) PLC प्रोग्रामिंग रिकॉर्ड को पेपर, टेप या कम्प्यूटर डिस्क में रखना चाहिए ताकि इन्हें भविष्य में कभी प्राकृतिक आपदा के बाद पुन: उपयोग में लाया जा सके।

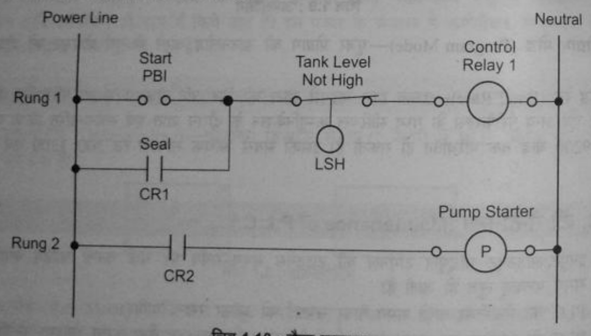
(viii) PLC को रिप्लेस करते समय पर्याप्त बैटरी बैकअप होना चाहिए। इसमें बैकअप के लिए लीथियम की बैटरी उपयोग में लानी चाहिए।

(ix) प्रत्येक PLC की लिखित चैक लिस्ट कन्ट्रोल शीट रखनी चाहिए जिससे कि उसकी अगली मेन्टीनेन्स प्रोसेस को आसानी से सम्पन्न किया जा सके।

(x) प्रत्येक एडीशनल PLC की मेन्टीनेन्स लॉग शीट रखनी चाहिए। यह शीट भविष्य में नई PLCs को परफॉर्मेन्स बेस पर खरीदने के लिए सहायक होती है।

## P.L.C. में उपयोग होने वाली विभिन्न प्रोग्रामिंग लैंग्वेज (Different Programming Languages used in P.L.C.)

**(i) लैडर डायग्राम लैंग्वेज** (Ladder Diagram Language)—लैडर डायग्राम (LAD) सबसे अधिक सामान्य प्रोग्रामेबल कन्ट्रोलर लैंग्वेज है। इसमें विभिन्न इन्स्ट्रक्शनों के सेट सम्मिलित रहते हैं जो सबसे अधिक बेसिक कन्ट्रोल फंक्शन्स, जैसे—रिले टाइप लॉजिक, टाइमिंग एवं काउन्टिंग तथा बेसिक मैप ऑपरेशन्स आदि परफॉर्म करते हैं। इस प्रकार से प्रोग्रामेबल लॉजिक मॉडल पर आधारित होकर इन्स्ट्रक्शन्स अन्य ऑपरेशन्स परफॉर्म करने के लिए विकसित किये जाते हैं। इसमें अतिक्ति फंक्शन, जैसे—एनालॉग कन्ट्रोल, डाटा मैनिपुलेशन, रिपोर्टिंग, कॉम्प्लेक्स, कन्ट्रोल लॉजिक आदि उपयोग मे लाये जाते हैं। LAD एक सिम्बॉलिक इन्स्ट्रक्शन सेट है जिसका उपयोग प्रोग्रामेबल कन्ट्रोल को निर्मित करने में किया जाता है। यह इन्स्ट्रक्शन रिले टाइप, टाइमर/काउन्टर, डाटा मैनिपुलेशन, अर्थमैटिक, डाटा ट्रांसफर एवं प्रोग्राम कन्ट्रोल आदि श्रेणियों से मिलकर बना होता है।




चित्र 1.10 : लैडर डायग्राम

LAD प्रोग्राम का मुख्य फंक्शन इनपुट्स की कन्डीशन्स पर आउटपुट को कन्ट्रोल करना होता है। इसमें एक लैडर लॉजिक रंग (rung) में इनपुट कन्डीशन्स का सेट सम्मिलित किया जाता है जिसको रिले कॉन्टेक्ट टाइप इन्स्ट्रक्शन्स द्वारा प्रदर्शित किया जाता है। क्वॉइल्स एवं कन्टेन्ट्स लैडर डायग्राम इन्स्ट्रक्शन सेट के बेसिक सिम्बल होते हैं जो दिये गये रंग मे प्रोग्राम्ड कन्टेन्ट सिम्बल्स व कन्डीशन्स रिप्रेजेन्ट करते हैं। इसमें सिम्बल निर्धारित करने के लिए मूल्यांकन करने की आवश्यकता होती है, कि आउटपुट को किस प्रकार कन्ट्रोल किया जाना है। सभी डिस्क्रीट आउटपुट क्वॉइल सिम्बल्स द्वारा रिप्रेजेन्ट किये जाते हैं। इस प्रकार से जब प्रोग्रामिंग की जाती है तब प्रत्येक कन्टेन्ट क्वॉइल को एक एड्रेस



नम्बर के साथ रैफर्ड किया जाता है जिसके द्वारा यह पहचान की जाती है कि किसका मूल्यांकन किया जाना है एवं क्या कन्ट्रोल करना है।

रंग कन्टेन्ट्स का फॉर्मेट, डिजायर्ड कन्ट्रोल लॉजिक पर निर्भर रहता है। कन्टेन्ट्स को सीरीज, पैरेलल अथवा सीरीज एवं पैरेलल के कॉन्फिग्युरेशन में स्थानान्तरित किया जाता है जिसकी आवश्यकता प्राप्त आउटपुट को कन्ट्रोल करने में पड़ती है। इस आउटपुट को सक्रिय अथवा उत्तेजित होने के लिए कन्टेन्ट्स का कम से कम एक पाथ बंद होना चाहिए। इस प्रकार जब कम से कम एक साथ में लॉजिक कन्टीन्यूटी रहती है तब रंग (rung) कन्डीशन को ऑन (on) माना जाता है। इस प्रकार यदि किसी भी पाथ में कन्टीन्यूटी नहीं है तब रंग कन्डीशन ऑफ (off) मानी जाती है। लैडर डायग्राम को चित्र 1.10 में दर्शाया गया है

**(ii) इन्स्ट्रक्शन लिस्ट लैंग्वेज** (Instruction List Language)—यह किसी भी माइक्रोप्रोसेसर/कम्प्यूटर की असेम्बली लैंग्वेज होती है। निमोनिक्स तथा विभिन्न ऑपरेशन परफॉर्म करने के लिए यह लैंग्वेज उपयोग में लायी जाती है। असेम्बली लैंग्वेज PLC के लॉजिक को दर्शाने के लिए उपयोग की जाती है। इस लैंग्वेज में उपयोग होने वाले प्रमुख इन्स्ट्रक्शन नीचे सारणी में दिये गये हैं—

| क्र०सं० | इन्स्ट्रक्शन्स (Instructions) | निमोनिक्स (Mnemonics) |
|---|---|---|
| (i) | इनपुट/आउटपुट | LD/OUT or OUT NOT STO |
| (ii) | अर्थमैटिक | ADD, SUB, MUL, DIV |
| (iii) | टाइमर | TIM |
| (iv) | काउन्टर | CTU, CTD |
| (v) | कम्पेयर | CMP |
| (vi) | लॉजिकल | AD, OR |

इस प्रकार से निमोनिक्स को निम्नलिखित प्रोग्राम द्वारा दर्शाया गया है—

```
LD A
AD B
OUT D
END
```

इसमें इनपुट डिवाइस *A* से इनपुट प्राप्त करती है तथा इनपुट *B* स्टेटस के साथ लॉजिकल AND ऑपरेशन परफॉर्म करती है। इस प्रकार से इस इनपुट ऑपरेशन को आउटपुट डिवाइस *D* पर प्राप्त किया जाता है।

**(iii) हायर लेवल लैंग्वेज** (Higher Level Language)—हायर लेवल लैंग्वेज एक कम्प्यूटर टाइप लैंग्वेज (CTL) है जिसका उपयोग अंग्रेजी के स्टेटमेन्ट्स तथा इन्स्ट्रक्शन्स को नियुक्त करने में उपयोग किया जाता है। यह लैंग्वेज सामान्यत: BASIC लैंग्वेज के समान होती है। विभिन्न प्रकार की PLC में BASIC लैंग्वेज का उपयोग किया जाता है। कुछ कम्प्यूटर टाइप लैंग्वेज BASIC से भी सरल तथा यूजर फ्रैंडली एवं ऑपरेटर ऑरियेन्टेड होती हैं इसलिए इस लैंग्वेज को ऑपरेटर द्वारा समझना आसान होता है।

LET, INPUT, READ तथा DATA आदि इन्स्ट्रक्शन्स लैडर कन्टेन्ट सिम्बल के समान होते हैं। LET इन्स्ट्रक्शन का उपयोग वेरियेबल को वेल्यू देने के लिए किया जाता है। इसमे CTL, READ इन्स्ट्रक्शन इनपुट के समान होते हैं जो IF एवं THEN इन्स्ट्रक्शन्स के साथ उपयोग किये जाते हैं।

**उदाहरण**—माना कि एक लैडर रंग को सिमुलेट करना है जिसमें एक N.O. (normally open) कन्टेन्ट (X) तथा जिसका आउटपुट (Y) है। जब N.O. कन्टेन्ट बंद हो जाता है तब आउटपुट उत्तेजित हो जाता है, जिसको निम्नलिखित प्रोग्राम से प्रदर्शित किया जा सकता है—

```
10      READ X
20      IF X = 0 THEN Y = 0
30      IF X = 1 THEN Y = 1
40      END
```

इसमें IF/THEN पेयर एक कन्डीशन का निर्माण करते हैं जिसके संतुष्ट हो जाने पर परिणाम एक विशिष्ट आउटपुट के रूप में प्राप्त होता है। DATA स्टेटमेन्ट प्रोग्राम में एक लाइन होती है जो प्रोग्राम में एक्जीक्यूशन के लिए उपयोग की जाती है। टाइमर ऑपरेशन को TIMER ON, TIMER OFF, TIMER STOP इन्स्ट्रक्शन द्वारा कन्ट्रोल किया जाता है। WAIT, FOR, STEP एवं NEXT इन्स्ट्रक्शन को काउन्टर के ऑपरेशन को सिमुलेट करने के लिए उपयोग किया जाता है। RUN तथा END एक्जीक्यूशन का उपयोग प्रोग्राम को प्रोपर एक्जीक्यूशन प्राप्त करने के लिए किया जाता है। इस प्रकार जब कीबोर्ड द्वारा RUN तथा ENTER टाइप किया जाता है तब प्रोग्राम पुनः एक्जीक्यूट हो जाता है।

## अभ्यासीय प्रश्न

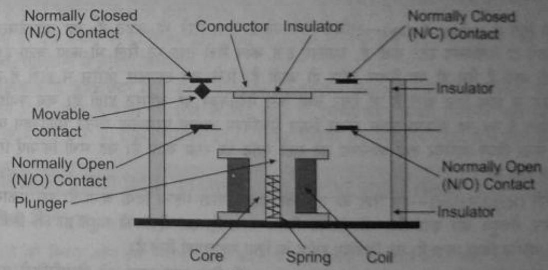
1. प्रोग्रामेबल लॉजिक कन्ट्रोलर क्या है? इसकी तकनिकी परिभाषा दीजिए।
2. PLC की हानियो को लिखिए।
3. PLC के लाभों को लिखिए।
4. PLC में उपयोग होने वाली प्रोग्रामिं लैंग्वेजों को समझाइए।
5. PLC के प्रकारों को समझाइए।
6. PLC के ब्लॉक डायग्राम को समझाइए।
7. PLC में उपयोग किये जाने वाले सामान्य टर्म्स को समझाइए।
8. PLC की विशेषताएँ क्या है? लिखिए।
9. PLC तथा कम्प्यूटर में अन्तर स्पष्ट कीजिए।
10. PLC मेन्टीनेन्स प्रोसीजर केचरणों को लिखिए।

❑

# 2
# इन्स्ट्रक्शन सेट (रिले, टाइमर्स तथा काउन्टर्स)
## [Instruction Set (Relay, Timers and Counters)]

## PLC हार्डवेयर में रिलेज (Relays in PLC Hardware)

प्रारम्भिक इलेक्ट्रिकल कन्ट्रोल सिस्टम का निर्माण मुख्य रूप से रिलेज एवं स्विचेस के द्वारा किया गया। स्विचेज एक जानी-पहचानी डिवाइस है। कभी-कभी यह रिलेज से सम्बन्ध व्यक्त नहीं करती है। रिलेज का उपयोग मुख्य रूप से पावर ट्रांसमिशन में पावर को ट्रांसमिट करने में किया जाता है। चित्र 2.1 में रिलेज का क्रास सेक्शनल डायग्राम दर्शाया गया है।

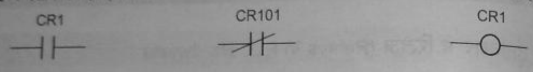


चित्र 2.1 : रिले

रिले अथवा कॉन्टेक्टर इलेक्ट्रोमैग्नेट क्वॉइल एवं कॉन्टेक्ट्स द्वारा निर्मित एक इलेक्ट्रोमैग्नेटिक डिवाइस है। इसमें मूवेबल कॉन्टेक्ट्स प्लंजर के साथ एक इन्स्यूलेटर के द्वारा जुड़े रहते हैं जो एक बोबिन के अन्तर्गत मूव करते हैं। इलेक्ट्रोमैग्नेट उत्पन्न करने के लिए कॉपर के तार को मुण्डली के रूप में बोबिन पर लपेटा जाता है। स्प्रिंग, प्लंजर को इलेक्ट्रोमैग्नेट से ऊपर तथा दूर रखती है। जब क्वॉइल द्वारा इलेक्ट्रिक करंट को पास करके इलेक्ट्रोमैग्नेट को सक्रिय किया जाता है तब मैग्नेटिक फील्ड प्लंजर को कोर की तरफ खींचती है जिसके द्वारा मूवेबल कॉन्टेक्ट नीचे की तरफ खिंचता है। जब कॉन्टेक्ट्स के दो फिक्स्ड पेयर इलेक्ट्रिकल इन्स्यूलेटर पर रिले के साथ फ्रेम किये जाते हैं तब मूवेबल कॉन्टेक्ट्स कोर की तरफ नहीं खिच रहा होता है। इसमे कन्डक्टर भौतिक रूप से कॉन्टेक्ट्स के अपर फिक्सड पेयर को स्पर्श करता है तथा वह कोर को अपनी ओर खींचता है जिससे वह फिक्स्ड कॉन्टेक्ट्स के लोअर पेयर को स्पर्श करता है। इस प्रकार से रिले फ्रेम से जुड़े हुए कॉन्टेक्ट्स के बहुत से पेयर हो सकते हैं। रिले क्वॉइल को दी गई पावर के परिणामस्वरूप कॉन्टेक्ट्स सक्रिय एवं निष्क्रिय होते हैं। इस प्रकार जब कोई क्वॉइल निष्क्रिय की जाती है तब मूवेबल कॉन्टेक्ट्स अपर फिक्स्ड कॉन्टेक्ट पेयर से कनेक्ट हो जाते हैं। इन फिक्स्ड कॉन्टेक्ट्स को सामान्य रूप से नॉर्मली

क्लोज्ड कॉन्टेक्ट्स कहा जाता है। जब भी रिलेज "पावर ऑफ" स्टेट में होती है तब इसे मूवेबल कॉन्टेक्ट्स ... कन्डक्टर कहा जाता है। इसी प्रकार मूवेबल कॉन्टेक्ट्स रिले क्वॉइल के निष्क्रिय होने पर लोअर फिक्स्ड कॉन्टेक्ट्स पेयर से कनेक्ट नहीं होते हैं। इन फिक्स्ड कॉन्टेक्ट्स को नॉर्मली ओपन कॉन्टेक्ट्स कहा जाता है। इस प्रकार से क्वॉइल के निष्क्रिय होने पर ओपन कॉन्टेक्ट्स OFF कहलाते हैं एवं क्वॉइल के निष्क्रिय होने पर ON कहलाते हैं। इस प्रकार क्लोज कॉन्टेक्ट्स क्वॉइल के निष्क्रिय होने पर ON कहलाते हैं एवं सक्रिय होने पर OFF कहलाते हैं। सामान्य रूप से डिजिटल लॉजिक से सम्बन्धित डिजाइनर्स नॉर्मली ओपन कॉन्टेक्ट्स पर नॉन-इन्वर्टिंग की तरह एवं नॉर्मली क्लोज्ड कॉन्टेक्ट्स पर इन्वर्टिंग की तरफ कार्य करते हैं।

इस प्रकार से रिलेज में यह याद रखना जरूरी होता है कि इलेक्ट्रिकल डायग्राम्स में उपयोग होने वाले सिम्बल्स, इलेक्ट्रॉनिक्स डायग्राम से भिन्न होते हैं। इलेक्ट्रिकल मशीन डायग्राम्स में सबसे अधिक उपयोग होने वाले सिम्बल्स चित्र 2.2 में दर्शाये गये हैं। इन सिम्बल्स को नॉर्मली ओपन, नॉर्मली कलोज्ड तथा क्वॉइल कहा जाता है।




(अ) नॉर्मली ओपन कॉन्टेक्ट (ब) नॉर्मली क्लोज्ड कॉन्टेक्ट (स) क्वॉइल

चित्र 2.2 : रिले

## रिले के प्रकार (Types of Relay)

(i) **लैचिंग रिले** (Latching Relay)—लैचिंग रिले को इम्पल्स रिले भी कहते हैं। यह बाइस्टेबल मोड में कार्य करती है तथा इसमें दो रिलेक्सिंग स्टेट होती हैं, इसलिए इन्हें कीप रिले तथा स्टे रिले भी कहा जाता है। इसमें जैसे ही करंट प्रवाहित की जाती है वैसे ही यह स्विच ऑफ हो जाती है। रिले जब लगातार प्रोसेस में होती है तब लास्ट स्टेट तक यह सोलेनॉइड के साथ प्राप्त होती है जो रेचट तथा केम मैकेनिज्म पर ऑपरेट होती है। जब क्वॉइल को रिलेक्स पॉइन्ट पर रखा जाता है तब यह प्रक्रिया ओवर सेन्टर स्प्रिंग मैकेनिज्म अथवा परमानेन्ट मैग्नेट मैकेन्जिम कहलाती है। इस प्रकार से ओवर सेन्टर स्प्रिंग आर्मेचर तथा कॉन्टेक्ट को सही स्पॉट पर रखा जाता है। यह सभी क्रियाएँ रिमेनेन्ट कोर की सहायता से सम्पन्न होती हैं।

(ii) **रीड रिले** (Read Relay)—इस रिले को कॉन्टेक्ट द्वारा ज्यादा महत्व दिया जाता है। इस प्रकार की रिले द्वारा वातावरण प्रोटेक्शन, वेक्यूम और इनर्ट गैस को प्रोटेक्ट किया जाता है। इस रिले को बहुत ही लो-स्विचिंग करंट तथा वोल्टेज रेटिंग पर ऑपरेट किया जाता है। यह स्विचिंग स्पीड के लिए महत्वपूर्ण रिले है।

(iii) **पोलराइज्ड रिले** (Polarized Relay)—इस प्रकार की रिले द्वारा स्वयं की सेन्सटिविटी पर ज्यादा महत्व दिया जाता है। इस रिले का उपयोग टेलीफोन के अविष्कार के समय से किया जा रहा है। यह टेलीफोन एक्सचेंज तथा टेलीग्राफिक डिस्टोर्शन में महत्वपूर्ण भूमिका अदा करती है। इस रिले की सेन्सटिविटी को आर्मेचर रिले के परमानेन्ट मैग्नेट तथा पोल के बीच एडजेस्ट करना बहुत आसान होता है।

(iv) **बुचहोल्ज रिले** (Buchholz Relay)—इस रिले का उपयोग सेफ्टी डिवाइस के रूप में किया जाता है। यह बड़े ऑयल-फिल्ड (filled) ट्रांसफॉर्मर में गैस की मात्रा कितनी है, यह जानकारी प्राप्त करती है। यह ट्राँसफॉर्मर में गैस के स्लो प्रोडक्शन तथा फास्ट प्रोडक्शन को सेन्स करती है तथा अन्य चेतावनियों को भी सेन्स करती है।

(v) **ओवरलोड प्रोटेक्शन रिले** (Overload Protection Relay)—किसी इलेक्ट्रिक मोटर को ओवर करंट तथा ओवरलोडिंग प्रोटेक्शन से डेमेज होने से बचाने के लिए इस रिले का उपयोग किया जाता है। इस प्रकार की रिले में हीटिंग एलीमेन्ट के सीरीज में होने पर मोटर की इन्टर्नल वाइंडिंग में केबिल कनेक्टिंग के कारण शॉर्ट सर्किट होता है। मोटर में बाइमैटेलिक स्ट्रिप तथा सोल्डर के मेल्ट हो जाने के कारण हीटिंग जनरेट होती है जो वातावरण को प्रभावित करती है। इस प्रकार से उस मोटर को वातावरण के अनुसार टेम्प्रेचर प्रोवाइड कराया जाता है।



(vi) **मर्करी वेटेड रिले** (Mercury Wetted Relay)—यह रिले, रीड रिले के समान ही होती है। इसमें गैस को इन्सर्ट किया जाता है तथा कॉन्टेन्ट को मर्करी से वेटेड किया जाता है। यह रिले बहुत ही सेन्सटिव तथा एक्सपेन्सिव होती है। इसको किसी ऑपरेशन के लिए लम्बवत् लगाया जाता है। इसमें बहुत ही लो-रजिस्टर पाया जाता है इसलिए इसे टाइमिंग एप्लीकेशन में उपयोग किया जाता है। टाइमिंग एप्लीकेशन के कारण इस रिले को अधिक बार उपयोग नहीं किया जा सकता है।

(vii) **मशीन टूल रिले** (Machine Tool Relay)—यह बहुत ही महत्वपूर्ण इन्डस्ट्रियल रिले है। इसका उपयोग मुख्य रूप से सभी प्रकार की मशीनों को कन्ट्रोल करने में किया जाता है। इसमें अधिक से अधिक कॉन्टेक्ट होते हैं जो क्वॉइल के साथ आसानी से रिप्लेस किये जा सकते हैं। यह रिले नॉर्मली ओपन तथा नॉर्मली क्लोज्ड कॉन्टेक्ट के लिए उपयोगी है। इस रिले को कन्ट्रोल पेनल के हिसाब से आसानी से सेटअप किया जा सकता है। इस रिले का उपयोग PLC का आविष्कार हो जाने के बाद भी किया जाता है।

(viii) **कॉन्टेक्टर रिले** (Contactor Relay)—इस प्रकार की रिले का उपयोग हैवी लोड में किया जाता है। इसको मुख्य रूप से इलेक्ट्रिक मोटर की स्विचिंग के लिए उपयोग में लाया जाता है। इसमें करंट की रेटिंग शून्य ऐम्पियर से 100 ऐम्पियर तक होती है। इस प्रकार की रिले का उपयोग रफ एरिया में किया जाता है। इस प्रकार की रिले द्वारा नॉइस उत्पन्न की जाती है इसलिए इस रिले का उपयोग उस क्षेत्र में नहीं किया जाता है जहाँ नॉइस एक समस्या है।

(ix) **सॉलिड स्टेट रिले** (Solid State Relay)—सॉलिड स्टेट रिले एक सॉलिड स्टेट इलेक्ट्रॉनिक कम्पोनेन्ट है जो इलेक्ट्रोमैग्नेटिक रिले के समान फंक्शन प्रदान करता है लेकिन इसमें एक भी मूविंग कम्पोनेन्ट तथा इन्क्रीजिंग रिलायबिलिटी नहीं होती है। इसके प्रत्येक ट्रांजिस्टर में स्मॉल वोल्टेज ड्रॉप होता है। यह वोल्टेज ड्रॉप एक निश्चित करंट मात्रा में होता है जिसे सॉलिड स्टेट के द्वारा हैंडिल किया जाता है। इस रिले के एक्रॉस मिनिमम वोल्टेज ड्रॉप ट्रांजिस्टर के बराबर होता है। इस प्रकार के ट्रांजिस्टर द्वारा 100 से 1200 ऐम्पियर तक करंट हैंडिल की जाती है तथा इस प्रकार के ट्रांजिस्टर को सिलिकॉन से बनाया जाता है।

(x) **सॉलिड स्टेट कॉन्टेक्टर रिले** (Solid State Contactor Relay)—यह रिले सॉलिड स्टेट तथा कॉन्टेक्टर रिले फीचर्स का मिला-जुला रूप है। इस रिले के कई लाभ हैं। इसको बहुत अच्छी हीटसिंक तथा ऑन-ऑफ साइकिल कॉन्टेक्ट के लिए बनाया जाता है। इसको मुख्यत: PLC, माइक्रोप्रोसेसर तथा माइक्रोकन्ट्रोलर से कन्ट्रोल किया जाता है।

## टाइम डिले रिले तथा प्रकार (Time Delay Relay and Types)

### टाइम डिले रिले (Time Delay Relay)

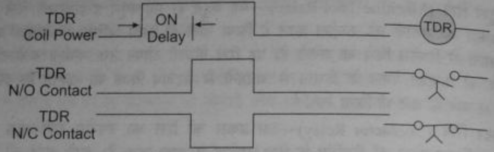
किसी रिले को स्विच ऑन तथा स्विच ऑफ करने में लगे समय को टाइम डिले कहा जाता है तथा इस प्रोसेस को जिस डिवाइस द्वारा कन्ट्रोल या सम्पन्न किया जाता है उसे टाइम डिले रिले कहा जाता है। इस रिले को टाइमिंग डिवाइस द्वारा ऑपरेट किया जाता है जिसमें माइक्रोप्रोसेसर अथवा माइक्रोकन्ट्रोलर का उपयोग किया जाता है। इस प्रकार की रिले को TDR से इंगित किया जाता है। इसमें यूजर के द्वारा एक निश्चित टाइम को किसी ऑपरेशन के लिए डिवाइस में सेट किया जाता है। इस रिले का उपयोग मुख्य रूप से इन्डस्ट्रियल ऑपरेशन में किया जाता है।

### टाइम डिले रिले के प्रकार (Types of Time Delay Relay)

टाइम डिले रिले के प्रमुख प्रकार निम्नलिखित हैं—

(i) **डिले ऑन टाइम रिले** (Delay ON Time Relay)—जब डिले ऑन टाइम को किसी सर्किट में इन्स्टॉल किया जाता है तब यूजर इच्छानुसार उसके टाइम डिले को प्राप्त करने के लिए रिले के टाइम को एडजेस्ट करता है। इस प्रकार के टाइम सेटिंग को प्रीसेट कहा जाता है चित्र 2.3 में डिले ऑन टाइम को प्रदर्शित किया गया है। इसमें टाइम डिले की प्रत्येक स्थिति को प्रदर्शित किया गया है जो रिले के विभिन्न टाइमिंग वेवफॉर्म को प्रदर्शित करती है। इसमें पावर देने के लिए पावर क्वॉइल का उपयोग किया जाता है जो रिले को इनर्जाइज्ड करती है तब रिले चलने लगती है।

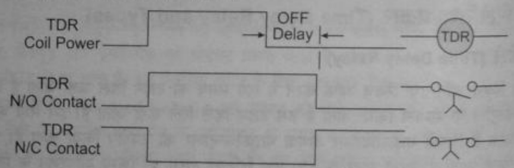
इसकी सहायता से किसी मैकेनिकल मोटर ड्राइव टाइमर अथवा इलेक्ट्रॉनिक टाइमर को टाइम डिले टाइमर की सहायता से ऑपरेट किया जाता है। इस प्रकार से जब क्वॉइल इनर्जाइज्ड हो जाती है तब इन्टर्नल टाइमर चलने की स्थिति में हो जाता है। जब टाइम वेल्यू के कारण टाइमर प्रीसेट स्थिति में पहुँचता है तब रिले इनर्जाइज्ड होती है। इस घटना को नार्मली ओपन कॉन्टेक्ट कहा जाता है तब रिले ऑफ होती है। जब रिले ऑन होती है तब इसे नॉर्मली क्लोज्ड कॉन्टेक्ट कहा जाता है।




चित्र 2.3 : डिले ऑन टाइम रिले

अतः इस प्रकार के टाइमर में यह याद रखना चाहिए कि जब रिले क्वॉइल से पावर हटा ली जाती है तब उसका कॉन्टेन्ट डी-इनर्जाइज्ड पोजीशन में पहुँच जाता है तथा टाइमर रीसेट होता है एवं रिले अगले टाइमर के लिए टाइम पावर देने के लिए तैयार हो जाती है। इसमें जब पावर के द्वारा पावर एप्लाई की जाती है तब इसका प्रीसेट टाइम रिले कॉन्टेन्ट पर कभी भी एक्टीवेट नहीं होता है। इस प्रकार से डिले ऑन रिले डिलेइंग टर्न ऑन इवेन्ट्स के लिए उपयोगी होता है, जैसे—जब एक मोटर के द्वारा किसी मशीन को स्टार्ट किया जाता है तब टर्न ऑन टाइम सभी कन्ट्रोल सिग्नलों को कुछ सेकण्ड के लिए डिसेबल करके रखता है जब तक कि मशीन पूर्ण स्पीड में ना आ जाये।

**(ii) डिले ऑफ टाइम रिले** (Delay OFF Time Relay)—चित्र 2.4 में डिले ऑफ टाइम रिले का चित्र प्रदर्शित किया गया है। इसमें क्षण भर के लिए रिले क्वॉइल पर पावर एप्लाई की जाती है तो कॉन्टेन्ट्स एक्टीवेट हो जाते हैं एवं नॉर्मली ओपन कॉन्टेक्ट्स क्लोज्ड हो जाता है तथा नॉर्मली क्लोज्ड कॉन्टेक्ट्स ओपन हो जाता है। इसमें टाइम डिले तब तक उपस्थित रहता है जब तक रिले स्विच ऑफ रहती है। इस प्रकार जब पावर को रिले क्वॉइल से हटा लिया जाता है तब कॉन्टेक्ट्स एक छोटे समय के लिए एक्टीवेट रहता है।




चित्र 2.4 : डिले ऑफ टाइम रिले

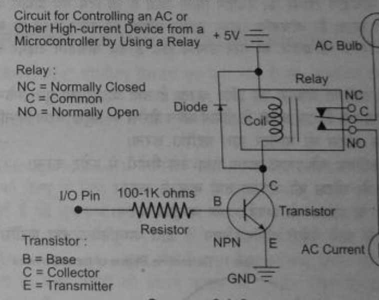
यदि रिले क्वॉइल टाइम आउट होने से पहले पुनः इनर्जाइज्ड हो जाती है तब टाइमर रीसेट हो जाता है तथा रिले इनर्जाइज्ड ही रहती है जब तक कि उस पर से पावर न हटा ली जाये। इस प्रकार से यह समय पुनः डिले ऑफ साइकिल को स्टार्ट कर देता है। डिले ऑफ टाइम डिले रिले का उपयोग किसी ऑपरेशन के ऑपरेशनल टाइम को बढ़ाने के लिए किया जाता है, जैसे—किसी फेन के द्वारा किसी मशीन को कूलिंग प्रदान करने के लिए लगातार ऑपरेट किया जाता है जब तक कि मशीन बन्द नहीं हो जाती है। इस प्रकार टाइम ऑफ सेन्सर द्वारा इस मोशन को सेन्स किया जाता है तथा स्विच को इमीडियेटली ऑन किया जाता है।



## रिले का सिद्धान्त (Principle of Relay)

जब किसी क्वॉइल को नॉन-इलेक्ट्रिकल मैटीरियल, जैसे—प्लास्टिक, पेपर पर लपेटा जाता है तब इसे एयर कोर सोलेनॉइड या साधारण सोलेनॉइड कहा जाता है। यदि सॉफ्ट आयरन को किसी क्वॉइल के अन्दर इन्सर्ट किया जाता है तब यह इलेक्ट्रोमैग्नेट की तरह कार्य करता है। इसको इलेक्ट्रोमैग्नेट रिले तथा अन्य इलेक्ट्रोमैकेनिकल डिवाइस, जैसे—इलेक्ट्रिक बैल एवं सर्किट ब्रेकर आदि में बेसिक कम्पोनेन्ट की तरह उपयोग किया जाता है।

जब क्वॉइल में करंट फ्लो की जाती है तब उसकी परिणामी मैग्नेटिक फील्ड मूविंग क्वॉइल पर मैकेनिकली लिंक रहती है। इस प्रकार मूवमेन्ट या तो बनाये जाते हैं या कॉन्टेन्ट के साथ ब्रेक किये जाते हैं। जब क्वॉइल में करंट स्विच ऑफ होती है तब आर्मेचर, मैग्नेटिक फोर्स के द्वारा अपनी रिलेक्स पोजीशन पर आ जाता है। सामान्यतः यह एक स्प्रिंग होती है लेकिन इसका गुरुत्वीय उपयोग मुख्य रूप से इन्डस्ट्रियल मोटर स्टार्टर में किया जाता है। मुख्यतः सभी रिले फास्ट ऑपरेशन के लिए मैन्यूफैक्चर किये जाते हैं। लो-वोल्टेज एप्लीकेशन में यह नॉइस को घटाती है तथा हाई वोल्टेज ऑपरेशन या हाई करंट एप्लीकेशन में यह आर्किंग को घटाती है।




चित्र 2.5 : रिले डिजाइन

चित्र 2.5 में रिले डिजाइन को दर्शाया गया है। यदि क्वॉइल DC करंट के साथ इनर्जाइज्ड होती है तब डायोड को क्वॉइल के एक्रॉस लगाया जाता है जो मैग्नेटिक फील्ड से ही एक्टीवेटशन के लिए इनर्जी को डेसीपेट करती है। यह स्पाइक वोल्टेज को डेमेज करती है तथा इसके द्वारा सर्किट कम्पोनेन्ट भी डेमेज हो जाता है। इसमें कुछ ऑटोमेटिव रिले पहले से ही डायोड को रखती हैं। इस प्रकार यदि क्वॉइल को AC करंट के साथ इनर्जाइज्ड के लिए डिजाइन किया जाता है तब एक कॉपर रिंग की सोलेनॉइड के अन्त में क्लेम्प किया जाता है। यह सेडिंग रिंग फेज करंट के द्वारा स्मॉल आउटपुट उत्पन्न करती है जो AC साइकिल में मिनिमम पुल ऑफ टाइम को बढ़ाती है। इलेक्ट्रोमैग्नेटिक डिजाइन की इस एनालॉजी के द्वारा एक सॉलिड स्टेट रिले थायरिस्टर या अन्य सॉलिड स्टेट स्विचिंग डिवाइस बनायी जाती है। इलेक्ट्रिकल आइसोलेशन प्राप्त करने के लिए लाइट इमीटिंग डायोड को फोटो ट्रांजिस्टर के साथ उपयोग किया जाता है।

इस प्रकार से रिलेज, स्विच्ड हैं जो टर्मनोलॉजी स्विच में उपयोग किये जाते हैं। एक रिले को एक या एक से अधिक पोल पर स्विच किया जा सकता है। इस प्रकार से क्वॉइल द्वारा निम्नलिखित तीन फंक्शन परफॉर्म किये जाते हैं—

(i) नॉर्मली क्लोज्ड कॉन्टेक्ट को जब सर्किट से कनेक्ट करते हैं तब रिले एक्टीवेट रहती है। यदि सर्किट डिसकनेक्ट होता है तब रिले एक्टीवेट नहीं रहती है। इसे "मेक" या A कॉन्टेक्ट कहा जाता है।

(ii) नॉर्मली क्लोज्ड कॉन्टेक्ट को जब सर्किट से डिसकनेक्ट करते हैं तब रिले एक्टीवेट रहती है। यदि

सर्किट कनेक्ट होता है तब रिले डिएक्टीवेट रहती है। इसे "ब्रेक" या B कॉन्टेक्ट कहा जाता है।

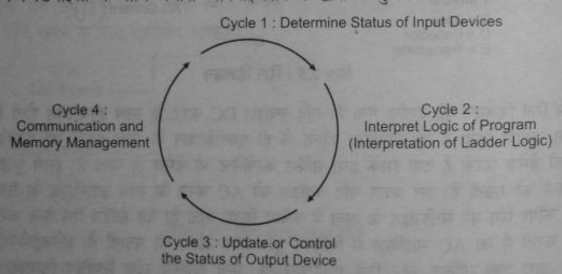
(iii) चेंज ओवर या डबल थ्रो कॉन्टेक्ट कन्ट्रोल में दो सर्किट होते हैं जिसमें एक नॉर्मली ओपन तथा दूसरा नार्मली क्लोज्ड के साथ कॉमन टर्मिनल से जुड़ा रहता है। इसे "ट्रांसफर" या C कॉन्टेक्ट कहा जाता है।

## प्रोग्राम स्कैन प्रोसेस (Process of Program Scan)

प्रोग्राम स्कैन प्रोग्रामेबल लॉजिक कन्ट्रोलर की एक आन्तरिक प्रोसेस है जिसमें सॉफ्टवेयर प्रोसेस का उपयोग किया जाता है। जब PLC को ऑन किया जाता है तब प्रोसेसर सभी आन्तरिक क्रियाओं को चैक करता है तथा यह निश्चित करता है कि PLC के सभी भाग कार्य करने की स्थिति में है। यह सभी प्रक्रियाएँ PLC में इन्स्टॉल किये गये सॉफ्टवेयर की सहायता से सम्पन्न होती हैं। इसमें सर्वप्रथम प्रोसेसर द्वारा PLC से जुड़े हुए सभी भागों को एक निर्देश भेजा जाता है जो उन निर्देशों को एक्जीक्यूट करता है। इस प्रकार से प्रोसेसर PLC में उपस्थित प्रोग्राम को स्कैन करना शुरू करता है। स्कैन प्रक्रिया में प्रोसेसर द्वारा वॉचडॉग टाइमर का उपयोग किया जाता है जो दिये गये प्रोग्राम की स्थिति को चैक करता है तथा आन्तरिक टेस्ट प्रदर्शित करता है। वॉचडॉग टाइमर प्रत्येक प्रोग्राम को स्कैन करने के बाद रीसेट होता है। इसमें यदि स्कैन के दौरान प्रोग्राम में किसी भी प्रकार की एरर उत्पन्न होती है तब वॉचडॉग टाइमर का टाइम आउट हो जाता है।

इस प्रकार यह प्रोग्राम में उपस्थित समस्या को इंगित करता है तथा प्रोग्राम को पुन: स्कैन करता है। चित्र 2.6 में प्रोग्राम की स्कैन प्रोसेस को दर्शाया गया है। इस प्रकार से प्रोग्राम स्कैन प्रोसेस के प्रमुख चरण निम्नलिखित हैं—

(i) इनपुट डिवाइस के स्टेटस को मॉनीटर द्वारा प्रदर्शित करना।
(ii) लैडर प्रोग्राम के लॉजिक को इन्ट्रप्ट करना तथा उसे मैमोरी में स्टोर करना।
(iii) आउटपुट डिवइस के स्टेटस को अपडेट तथा कन्ट्रोल करना।
(iv) आउटपुट डिवाइस के स्टेटस को अपडेट तथा कन्ट्रोल करना।
(v) विभिन्न डिवाइसों के साथ मैमोरी ऑर्गेनाइजेशन के द्वारा कम्युनिकेट तथा पार्टीसिपेट करना।




**चित्र 2.6 : प्रोग्राम स्कैन प्रोसेस**

अत: इस प्रकार यह कहा जा सकता है कि प्रोग्राम स्कैन प्रोसेस विभिन्न चरणों में सम्पन्न होती है। इसमें मुख्य रूप से AB, PLC का उपयोग किया जाता है जो यूजर प्रोग्राम को प्रोसेस स्कैन के माध्यम से बायें से दायें प्रोसेस करता है तथा स्कैनिंग ऊपर से नीचे की ओर करता है। इसमें प्रोसेस प्रक्रिया एक चक्र के रूप में सम्पन्न होती है।

## मैमोरी ऑर्गेनाइजेशन (Memory Organisation)

मैमोरी सिस्टम ROM तथा RAM से मिलकर बना होता है। ROM मैमोरी प्रोग्राम से सम्बन्धित सूचनाओं को



रखती है तथा यह CPU को लैडर लॉजिक प्रोग्राम के समय प्राप्त कराती है। इसमें RAM मैमोरी द्वारा लैडर लॉजिक प्रोग्राम को स्टोर किया जाता है। इस प्रकार से RAM मैमोरी को लैडर प्रोग्राम को कॉल करने के लिए हमेशा ऑन पोजीशन में रखा जाता है। एक प्रोसेसर में हजारों मैमोरी लोकेशन होते हैं जो सूचनाओं को स्टोर करके रखते हैं जिन्हें वर्ड या रजिस्टर के नाम से जाना जाता है। मैमोरी ऑर्गेनाइजेशन में मुख्य रूप से 8, 16 या 32 बिट वर्ड का उपयोग किया जाता है। इसमें सूचनाओं को 0 तथा 1 के रूप में स्टोर किया जाता है जिसमें 1, ON स्थिति को प्रदर्शित करता है तथा 0, OFF स्थिति को प्रदर्शित करता है। ये सभी प्रकार के बिट वर्ड डाटा फाइल में स्टोर किये जाते हैं।

डाटा फाइल में तीन प्रकार के एलीमेन्ट उपयोग किये जाते हैं जिसमें कुछ एलीमेन्ट एक ही वर्ड के होते हैं। जब वन वर्ड वेल्यू को मैमोरी रेन्ज – 32768 से + 32767 में स्टोर किया जाता है तब टू वर्ड फ्लोटिंग-पॉइन्ट फाइल मैमोरी का उपयोग करता है। इस प्रकार से काउन्टर तथा टाइमर में थ्री वर्ड मैमोरी का उपयोग किया जाता है। इस प्रकार एलीगेन्ट प्रीएजाइन, स्टेटस बिट तथा एक्यूमुलेटेड वर्ड प्रति वेल्यू के रूप में स्टोर रहते हैं।

AB मैमोरी ऑर्गेनाइजेशन में ऑक्टल नम्बर बिट 00 से 07 बिट पॉजीशन का उपयोग किया जाता है जो बिट की पोजीशन को एड्रेस करके रखता है। इसमें 8 तथा 9 नम्बरों का उपयोग नहीं किया जाता है।

## AB मैमोरी ऑर्गेनाइजेशन (AB Memory Organization)

AB मैमोरी ऑर्गेनाइजेशन, PLC प्रोसेसिंग सिस्टम का एक भाग है जो आन्तरिक क्रियाओं को एक निश्चित क्रम में प्रदर्शित करता है। AB मैमोरी ऑर्गेनाइजेशन में भी RAM तथा ROM मैमोरी पायी जाती है जो विभिन्न प्रकार के वर्ड तथा रजिस्टर को एक निश्चित एलोकेशन में स्टोर करके रखती है। AB मैमोरी ऑर्गेनाइजेशन को निम्नलिखित दो भागों में बाँटा जा सकता है—

**(i) स्टोरेज मैमोरी** (Storage Memory)—स्टोरेज मैमोरी, इन्फॉर्मेशन को इन्टर्नल रिले स्टेटस में प्रीएसाइन वेल्यू तथा मैथमैटिकल फंक्शन की वेल्यू के लिए स्टेटस को I/O डिवाइस में स्टोर करके रखता है। यह प्रक्रिया डाटा टेबिल या रजिस्टर टेबिल कहलाती है जो सूचनाओं को स्टेटस एवं नम्बर के फॉर्म में स्टोर करके रखती है। इसमें स्टेटस ON तथा OFF के फॉर्म में स्टोर किये जाते हैं तथा नम्बर 0 से 1 के फॉर्म में मैमोरी के यूनिक बिट में स्टोर किये जाते हैं।

**(ii) यूजर मैमोरी** (User Memory)—यूजर मैमोरी द्वारा लैडर लॉजिक प्रोग्राम को प्राप्त किया जाता है तथा उसे प्रोग्राम में स्टोर किया जाता है। यूजर मैमोरी को प्रोग्राम फाइल या रजिस्टर टेबिल के द्वारा बनाया जाता है जो सम्पूर्ण ऑपरेशन को होल्ड करके रखता है।

इस प्रकार से AB मैमोरी ऑर्गेनाइजेशन को PLC टेबिल में स्टोरेज मैमोरी या डाटा फाइल के रूप में प्रदर्शित किया गया है।

टेबिल में अधिकतम एलीमेन्ट, फाइल टाइप, फाइल डिजाइनेशन, फाइल नम्बर तथा वर्ड प्रति एलीमेन्ट दर्शाये गये हैं। इन सभी को नीचे दी गई टेबिल में दर्शाया गया है—

| क्र० सं० | एलीमेन्ट की अधिकतम संख्या | फाइल टाइम | फाइल डिजाइनेशन | फाइल नम्बर | वर्ड/एलीमेन्ट |
|---|---|---|---|---|---|
| (i) | 32 | Output image | O | 0 | 1 |
| (ii) | 32 | Input image | I | 1 | 1 |
| (iii) | 1000 | Status | S | 2 | 1 |
| (iv) | 1000 | Bit | B | 3 | 1 (16) bit |
| (v) | 1000 | Timer | T | 4 | 3 |
| (vi) | 1000 | Counter | C | 5 | 3 |

| | | | | | |
|---|---|---|---|---|---|
| (vii) | 1000 | Control | R | 6 | 3 |
| (viii) | 1000 | Integer | N | 7 | 1 |
| (ix) | 1000 | Floating Point | F | 8 | 2 |

## स्टोरेज मैमोरी सेक्शन (Storage Memory Section)

स्टोरेज मैमोरी सेक्शन, मैमोरी ऑर्गेनाइजेशन पर आधारित है। इसमें स्टोर किये जाने वाले डाटा की स्थिति को ए निश्चित फाइल टाइप में स्टोर किया जाता है। स्टोरेज मैमोरी सेक्शन के प्रमुख फाइल टाइप निम्नलिखित हैं—

**(i) आउटपुट इमेज** (Output Image)—इसे फाइल O नाम से भी जाना जाता है। इसमें 32 वर्ड की मैमोरी हो है जो 16-बिट में विभाजित रहती है। इसमें आउटपुट (ON/OFF) का स्टेटस प्रत्येक स्कैन के बाद अपडेट होता है।

**(ii) इनपुट इमेज** (Input Image)—इसे फाइल 1 नाम से जाना जाता है। इसमें 32 वर्ड साइज की मैमोरी पा जाती है। इसमें इनपुट डिवाइस का स्टेटस प्रत्येक स्कैन के बाद अपडेट होता है।

**(iii) स्टेटस** (Status)—यह प्रोसेस ऑपरेशन की इन्फॉर्मेशन को, स्टेटस में स्टोर करके रखता है तथा ट्रबलशूटि के प्रोग्राम कोड को सर्च करता है। इसे S फाइल के नाम से जाना जाता है।

**(iv) बिट** (Bit)—इसका उपयोग इन्टर्नल या डमी रिले में किया जाता है तथा इसकी रेन्ज 1 से 100 वर्ड त होती है। इस फाइल का प्रत्येक एड्रेस B3 से स्टार्ट होता है। इसे फाइल B भी कहा जाता है।

**(v) टाइमर** (Timer)—इसमें मैमोरी के तीन वर्ड जिसमें स्टोर स्टेटस बिट, प्रीएसाइन वेल्यू तथा एक्यूमुलेटि वेल्यू आदि का उपयोग किया जाता है। इस फाइल का प्रत्येक एड्रेस T4 से स्टार्ट होता है। इसे T फाइल भी कहा जा है।

**(vi) काउन्टर** (Counter)—यह टाइमर की तरह ही उपयोग में लाया जाता है। इसमें भी तीन वर्ड होते हैं। इ फाइल का प्रत्येक एड्रेस C5 से स्टार्ट होता है। इसे C फाइल के नाम से जाना जाता है।

**(vii) कन्ट्रोल** (Control)—इसमें सिक्वेन्स तथा शिफ्ट रजिस्टर का उपयोग किया जाता है। इसमें प्रत्येक एड्रे फाइल R6 से स्टार्ट होती है। कन्ट्रोल फाइल में कन्ट्रोल वर्ड लेन्थ, पोजीशन आदि वर्ड्स का उपयोग किया जाता है। इ R फाइल कहा जाता है।

**(viii) इन्जीजर** (Integer)—इस प्रकार की फाइल में न्यूमेरिकल वेल्यू के सभी नम्बर स्टारे करके रखे जाते हैं इस फाइल को N फाइल से जाना जाता है।

**(ix) फ्लोटिंग** (Floating)—इस फाइल में डेसीमल या फ्लोटिंग-पॉइन्ट नम्बर स्टोर किये जाते हैं। इसे F फाइ से जाना जाता है।

**(x) फाइल 9-999 (File 9-999)**—इस फाइल टाइप को एसाइन करता है तथा आवश्यकता पड़ने पर फाइल व साइज को बढ़ाया भी जाता है।

अत: इस प्रकार से यह कहा जा सकता है कि बिट, इन्टीजर या फ्लोटिंग-पॉइन्ट टाइप फाइल स्टोर की जाती इस प्रकार यूजर आवश्यकतानुसार फाइल को सिलेक्ट करता है। यदि यूजर एक से अधिक डाटा फाइल के स्टेटस लिए किसी अन्य स्टेटस या डाटा फाइल का उपयोग करता है तब यूजर के द्वारा फ्लोटिंग-पॉइन्ट के लार्ज तथा स्मॉ नम्बर को उपयोग में लाया जाता है।

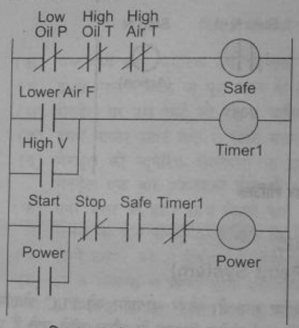
## PLC का लैडर लॉजिक (Ladder Logic of PLC)

लैडर लॉजिक एक प्रोग्रामिंग लैंग्वेज है जो ग्राफिकल डायग्राम पर आधारित है, जिसे रिले लॉजिक हार्डवेयर सर्कि डायग्राम की सहायता से रिप्रेजेन्ट करती है। इसका प्राथमिक उपयोग इन्डस्ट्रियल एप्लीकेशन प्रोग्राम लॉजिक कन्ट्रोलर किया जाता है जो सम्बन्धित सॉफ्टवेयर उपलब्ध कराता है। PLC लैडर लॉजिक कम्प्यूटर पर आधारित है जो कम्प्यू

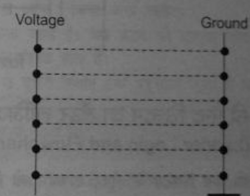


से सम्बन्धित मशीन लैंग्वेज तथा प्रोग्रामिंग लैंग्वेज को उपयोग में लाता है। PLC में मुख्य रूप से मशीन लैंग्वेज तथा लैडर लैंग्वेज का उपयोग किया जाता है। लैडर लॉजिक डायग्राम इलेक्ट्रोमैग्नेटिक रिले को लैडर डायग्राम में एक रिले टेक्नोलॉजी की सहायता से उपयोग करता है। रिले सर्किट को लैडर लॉजिक डायग्राम के फॉर्म में बनाया जाता है। इस प्रकार से यदि रिले को सॉलिड स्टेट स्विच के रूप में प्रोग्राम किया जाता है तब यह प्रोग्रामेबल लॉजिक के कॉम्बीनेशनल सर्किट या सिक्वेन्स नेटवर्क की तरह रीप्लेस किया जाता है।

लैडर लॉजिक के कॉम्बीनेशनल डायग्राम को चित्र 2.7 में प्रदर्शित किया गया है। इस वायरिंग डायग्राम में विभिन्न कम्पोनेन्ट जैसे—स्विच, रिले, मोटर्स आदि को दर्शाया गया है। इस प्रकार के कम्पोनेन्ट के इन्टरकनेक्शन का उपयोग इलेक्ट्रिशियन द्वारा कन्ट्रोल पेनल की वायरिंग के समय किया जाता है। इस प्रकार से लैडर डायग्राम द्वारा कन्ट्रोल सर्किट के डायग्राम को लम्बवत् लैडर के रंग्स (rungs) को लाइन द्वारा प्रदर्शित किया गया है। इसमें प्रतयेक कनेक्शन का परिणामी तथा उसका ऑपरेशन ज्ञात किया जाता है। लैडर डायग्राम का आधार दो लम्बवत् लाइनों पर टिका होता है जिसमें एक वोल्टेज सोर्स से कनेक्ट रहती है तथा दूसरी ग्राउण्ड से कनेक्ट रहती है जिसे लैडर लॉजिक का फ्रेमवर्क लैडर डायग्राम कहते हैं, जिसे चित्र 2.8 में दर्शाया गया है।



चित्र 2.7 : लैडर लॉजिक डायग्राम



चित्र 2.8 : लैडर लॉजिक फ्रेमवर्क

लैडर लॉजिक डायग्राम के कन्ट्रोल सिक्वेन्स में विभिन्स स्टेप्स तथा स्टेट का उपयोग किया जाता है जो प्रतयेक स्टेट तथा स्टेप को एक निश्चित क्रिया से कन्ट्रोल करता है। इस प्रकार एक या एक से अधिक स्थिति पर यह स्टेट एक्टिव होती है तथा अन्य स्टेट उसकी अलग स्थिति पर निर्भर करती है। लैडर लॉजिक की स्थिति प्रोसेस या कन्ट्रोल सिस्टम पर निर्भर करती है। लैडर लॉजिक डायग्राम के सिक्वेन्स का स्ट्रैक्चर चित्र 2.7 मे प्रदर्शित किया गया है। इस प्रकार के लैडर लॉजिक के एक्जीक्यूशन के दौरान जम्प तथा रिटर्न प्रोसेस का उपयोग किया जाता है जो स्टेप 1 से लास्ट स्टेप तक एक्जीक्यूट करती है। स्टेप N को चित्र 2.9 में दर्शाया गया है, जो निम्नलिखित प्रकार के स्टेप प्रदर्शित करता है–

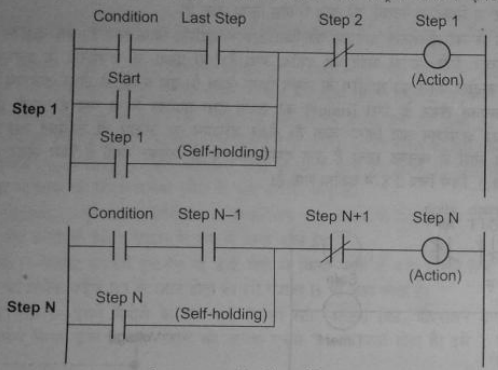
(i) जब स्टेप N + 1 एक्टिव नहीं होती है तब सभी स्टेप नॉर्मली क्लोज कॉन्टेक्ट में होते हैं तथा एक समय में एक स्टेप या स्टेट एक्टिव होती है।

(ii) जब स्टेप N – 1 एक्टिव होती है तब सभी स्टेप नॉर्मली ओपन कॉन्टेक्ट में होते हैं तथा पिछली स्टेप का सिक्वेन्स 1, 2, ... N – 1, N, N + 1 ... में होती है।

(iii) इस प्रकार अगर कन्डीशन सन्तुष्ट होती है तब यह प्रोसेस कन्ट्रोल सिस्टम को पहले की स्थिति को रिस्पॉन्स करने की अनुमति प्रदान करती है।

इस प्रकार से जब N स्टेप एक्टीवेट होता है तो यह निम्नलिखित स्टेप प्रदर्शित करता है—

(i) पिछले स्टेप को डिएक्टीवेट करता है जिसमें N + 1 स्टेप नॉर्मली क्लोज्ड से ओपन में आ जाती है।
(ii) स्टेप N के द्वारा लैच को एक्टिव स्टेट में रखता है।
(iii) स्टेप N के एक या अधिक स्टेप एक्टिव होने से प्रोसेस और कन्ट्रोल सिस्टम एक्टिव होता है।




चित्र 2.9 : N स्टेप लैडर लॉजिक

## स्प्रे पेन्ट सिस्टम का लैडर लॉजिक तथा फ्लोचार्ट (Ladder Logic and Flowchart of Spray Paint System)

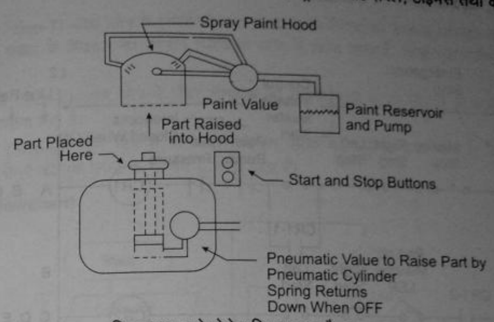
स्प्रे पेन्ट सिस्टम के लैडर डायग्राम को चित्र 2.11 में दर्शाया गया है। लैडर डायग्राम को PLC प्रोग्रामिंग की सहायता से बनाया जाता है। इस प्रकार लैडर डायग्राम के द्वारा बड़ी-बड़ी प्रोसेस सिस्टम के स्कैच खींचे जाते हैं एवं इस स्कैच की सहायता से विभिन्न प्रकार के कन्ट्रोल सिस्टमों को आपस में जोड़ा जाता है। स्प्रे पेन्ट प्रोसेस प्लानिंग सिस्टम के प्रमुख स्टेप निम्नलिखित हैं—

**(i) स्टेप-1** (Step-1)—इस स्टेप में स्प्रे पेन्टिंग पार्ट सिस्टम को सेटअप करते हैं। किसी भी पार्ट को मेनड्रेल (मेनड्रेल एक सॉफ्ट वेयर है जो वर्कपीस के दौरान सिस्टम में इन्सर्ट की जाती है) पर प्लेस किया जाता है। इसमें किसी पार्ट पर स्प्रे करने के लिए उस पार्ट को 6 सेकण्ड तक मेनड्रेल के सामने रखा जाता है तथा इसके बाद उस पार्ट को हटा दिया जाता है। इस प्रकार से मेनड्रेल अपनी ऑरिजनल पोजीशन पर वापस आ जाता है। इस प्रकार के स्प्रे प्रोसेस सिस्टम में न्यूमेटिक (pneumatic) सिलेण्डर का उपयोग किया जाता है। न्यूमेटिक सिस्टम में, पावर को प्रेशर के माध्यम से ट्रांसफर किया जाता है। गैस, प्रेसर पर आधारित रहती है जो न्यूमेटिक एक्च्यूयेटर से एक निश्चित प्रोसेस को सम्पन्न करती है। इस प्रकार से इस प्रक्रिया में मेन्यूफैक्चरिंग रोबोट सिस्टम का उपयोग किया जाता है।

**(ii) स्टेप-2** (Step-2)—इस स्टेप में स्प्रे पेन्ट के प्रोसेस ऑपरेशन का स्कैच डायग्राम बनाया जाता है जो चित्र 2.10 में दर्शाया गया है।

**(iii) स्टेप-3** (Step-3)—इस स्टेप में प्रोसेस सिस्टम की लिखित लिस्ट बनायी जाती है। इसकी प्रत्येक लाइन में इतना स्पेश दिया जाता है ताकि प्रोसेस सिस्टम के दौरान की गयी खोज को उस प्रोसेस में जोड़ा जा सके। इस प्रोसेस के प्रमुख बिन्दु निम्नलिखित हैं—






चित्र 2.10 : स्प्रे प्रोसेस सिस्टम का स्कैच डायग्राम

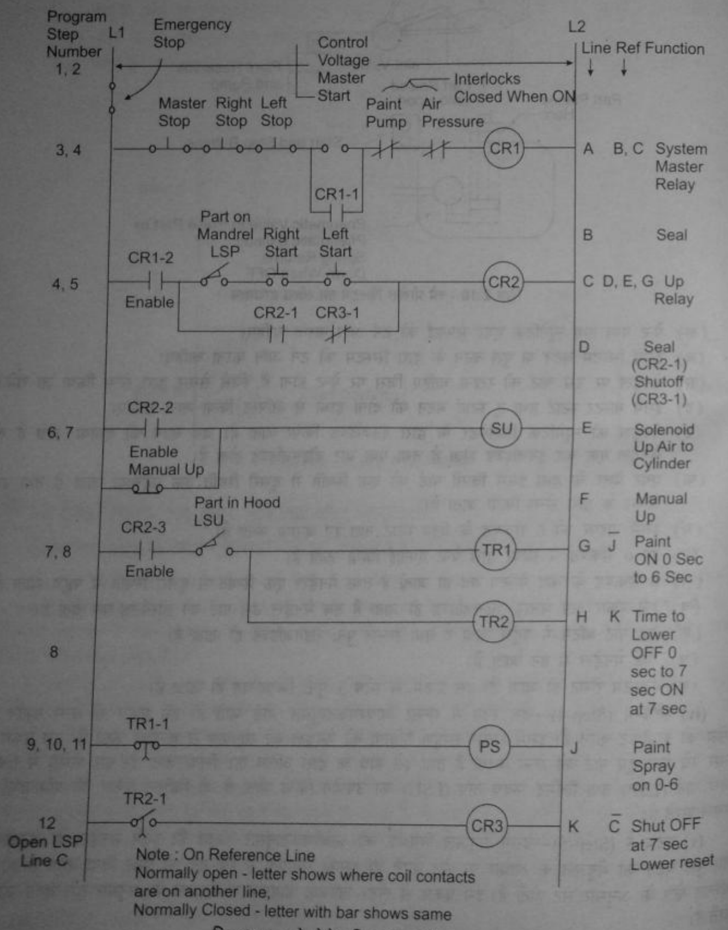
(अ) पेन्ट पम्प तथा न्यूमेटिक एयर सप्लाई को टर्न ऑन करना चाहिए।
(ब) इसमें सिस्टम बटन या पुश बटन के द्वारा सिस्टम को टर्न ऑन करना चाहिए।
(स) मेनड्रेल पर उस चार्ट को रखना चाहिए जिस पर पेन्ट होना है, जिसे सेन्सर द्वारा सेन्स किया जा सके।
(द) इसमें मास्टर स्टार्ट तथा टू स्टार्ट बटन को दोनों हाथों से ऑपरेट किया जाना चाहिए।
(इ) मेनड्रेल को न्यूमेटिक सिलेण्डर के द्वारा इनर्जाइज्ड किया जाता है। जब बटन को दबाया जाता है तो मेनड्रेल एक बार इनर्जाइज्ड होता है तथा एक बार डीइनर्जाइज्ड होता है।
(फ) एयर प्रेशर के द्वारा इसमें किसी पार्ट को एक स्थिति से दूसरी स्थिति तक पहुँचाया जाता है तथा इसे सेन्सर के द्वारा सेन्स किया जाता है।
(य) इसमें टाइमर को 6 सेकण्ड के लिए स्टार्ट तथा रन कराया जाता है।
(र) इन 6 सेकण्ड में स्प्रेयर द्वारा पेन्ट एप्लाई किया जाता है।
(ल) 6 सेकण्ड के बाद पेन्टिंग बंद हो जाती है तथा मेनड्रेल एक स्थिति से दूसरी स्थिति में पहुँच जाता है।
(व) इस प्रकार अप सेन्सर डीइनर्जाइज्ड हो जाता है तब मेनड्रेल उस पार्ट को डीस्कैन्ड कर देता है।
(श) यह पार्ट बॉटम में पहुँच जाता है तथा सेन्सर पुनः रीइनर्जाइज्ड हो जाता है।
(ष) पार्ट मेनड्रेल से हट जाता है।
(ह) सिस्टम रीसेट हो जाता है। इस प्रकार से स्टेप 3 पुनः क्रियान्वित हो जाता है।

**(iv) स्टेप-4** (Step-4)—इस स्टेप में सेन्सर आवश्यकतानुसार जोड़े जाते हैं। इस प्रकार से सेन्स मशीन की प्रोसेस को इन्डीकेट करते हैं। इसमें सेन्सर सम्पूर्ण प्रक्रिया को मेनड्रेल की सहायता से कन्ट्रोल करते हैं। इस प्रकार से सेन्सर स्प्रे पेन्ट हुए पार्ट को सेन्स करता है तथा इसे हाथ के द्वारा अलग कर लिया जाता है। इस प्रोसेस में लिमिट स्विच पार्ट (LSP) तथा लिमिट स्विच अप (LSU) का उपयोग किया जाता है जो विभिन्न प्रकार की प्रक्रियाओं को सम्पन्न करते हैं।

**(v) स्टेप-5** (Step-5)—इसमें मैनुअल नियंत्रक को आवश्यकतानुसार जोड़ते हैं। इसमें मेनड्रेल के सेटअप के लिए पुश बटन को मैनुअल के आधार पर सेट करते हैं। इसकी सहायता से स्प्रे-गन को प्रेशर दिया जाता है जो एक निश्चित क्षेत्र के अनुसार सेट होती है। इस प्रकार से लैडर लॉजिक डायग्राम में मैनुअल के अनुसार पुश बटन को सेट करते हैं।

**(vi) स्टेप-6** (Step-6)—इस स्टेप में मशीन ऑपरेटर की सुरक्षा को जोड़ा जाता है। इस ऑपरेशन में ऑपरेटर दोनों हाथों से किसी प्रोसेस को कन्ट्रोल करता है। इसमें दो पुश बटन उपयोग में लाये जाते हैं जो किसी प्रोसेस को स्टार्ट

या स्टॉप करते हैं।




चित्र 2.11 : स्प्रे प्रोसेस सिस्टम का लैडर डायग्राम

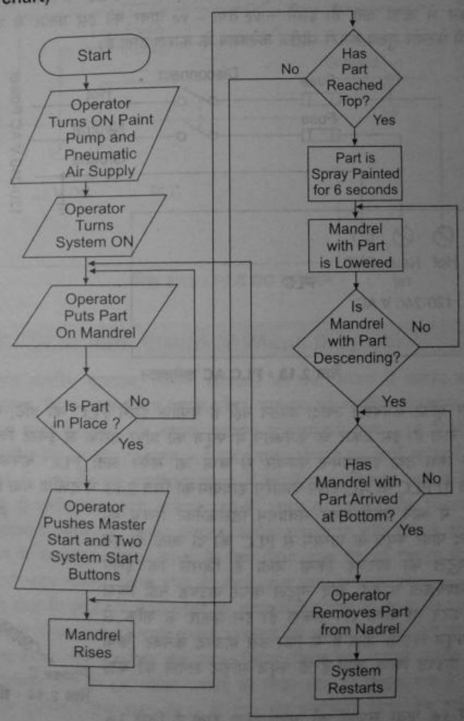


**(vii) स्टेप-7** (Step-7)—इस स्टेप में इमरजेन्सी तथा मास्टर स्टॉप स्विच का उपयोग ऑपरेटर की सुरक्षा के लिए किया जाता है। इस प्रकार के स्विचेस का उपयोग इमरजेन्सी स्टॉप में किया जाता है, जिन्हें एक एडीशनल स्टेप के रूप में जोड़ा जाता है।

**(viii) स्टेप-8** (Step-8)—इस स्टेप में लैडर डायग्राम को बनाया जाता है जो स्टेप 1 से 6 पर निर्भर रहता है। इसे चित्र 2.11 में दर्शाया गया है।

**(ix) स्टेप-9** (Step-9)—इस स्टेप में पोटेंशियल प्रोबल्म एरिया को प्रोसेस किया जाता है। इस प्रकार से इसमें सभी प्रकार की इमरजेन्सी को एक लिस्ट के माध्यम से जोड़ा जाता है।

## फ्लोचार्ट (Flowchart)



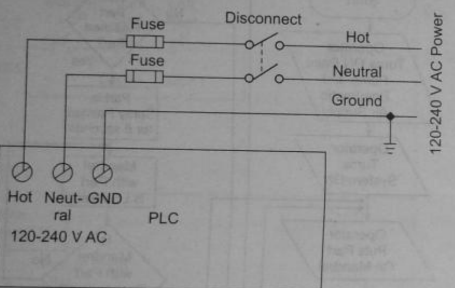

चित्र 2.12 : फ्लोचार्ट

फ्लोचार्ट के माध्यम से किसी लम्बी प्रोसेस प्रक्रिया को एक चार्ट के माध्यम से समझा जाता है। इस प्रकार के फ्लोचार्ट में प्रोसेस से सम्बन्धित लिस्ट तैयार की जाती है जिसमे विभिन्न प्रकार के सिम्बल्स उपयोग किये जाते हैं। स्प्रे

प्रोसेस सिस्टम को फ्लोचार्ट 2.12 में दर्शाया गया है। इस फ्लोचार्ट में स्प्रे प्रोसेस की सभी स्टेप को दर्शाया गया जो एक ऑपरेशन में उपयोग की जाती हैं।

## PLC कनेक्शन (PLC Connection)

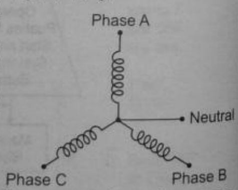
PLC कनेक्शन में मुख्य रूप से AC तथा DC पावर सप्लाई का उपयोग किया जाता है जो हॉट, न्यूट्रल तथा ग्राउण्ड कनेक्शन के माध्यम से आपस में जुड़े रहते हैं। PLC में पावर की आवश्यकता मॉडल के अनुसार होती है। PLC को 24 V AC, 120 V AC तथा 240 V AC पर PLC को ऑपरेट करते हैं। PLC पावर कनेक्शन में मुख्य रूप से DC प्रकार की पावर सप्लाई की आवश्यकता होती है। इस प्रकार के PLC कनेक्शन में + ve तथा – ve वायर को हॉट तथा न्यूट्रल प्रकार से जोड़ा जाता है। इसमें + ve तथा – ve वायर को इस प्रकार से जोड़ा जाता है कि शॉर्ट सर्किट न हो पाये। PLC में फेल्योर मुख्य रूप से सीरीज कनेक्शन के कारण होता है।




चित्र 2.13 : PLC AC कनेक्शन

PLC की AC पावर यूनिट कनेक्शन ज्यादा कठिन नहीं है क्योंकि इसमें PLC को हॉट, न्यूट्रल तथा ग्राउण्ड की सहायता से कनेक्ट किया जाता है। इस प्रकार के कनेक्शन में फ्यूज को प्रॉपर तरीके से इन्सर्ट किया जाता है जिससे कि शॉर्ट सर्किट, एक्सीडेन्टल केस तथा इक्यूपमेन्ट फेल्योर से बचा जा सके। अत: PLC कनेक्शन में प्रत्येक फ्यूजिंग सिस्टम को देखा जा सकता है। PLC के AC टाइप वायरिंग डायग्राम को चित्र 2.13 में दर्शाया गया है।

इस प्रकार से PLC में आने वाली पावर सर्वप्रथम डिसकनेक्ट स्विच को कनेक्ट करती है इसके बाद पावर फ्यूज के माध्यम से PLC को दी जाती है। इस प्रकार के सिस्टम में न्यूट्रल को ग्राउण्ड किया जाता है जिससे कि इसमें डिस्कनेक्ट स्विच की आवश्यकता न पड़े। यदि न्यूट्रल वायर ग्राउण्ड नहीं किया जाता हैं तब इसको टच करने पर शॉक लग सकता है। इस प्रकार के शॉक से बचने के लिए न्यूट्रल में फ्यूज लगाया जाता है या फिर उसे ग्राउण्ड कनेक्ट किया जाता है। यदि न्यूट्रल को ग्राउण्ड किया जाता है तो फ्यूज सर्किट लगाने की कोई आवश्यकता नहीं होती है।



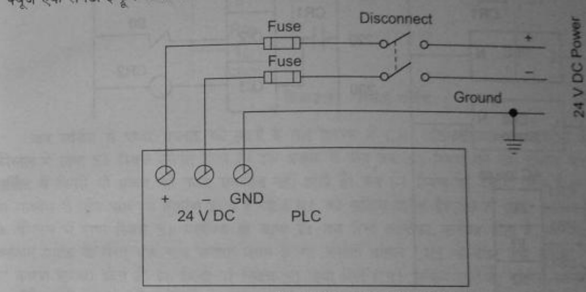

चित्र 2.14 : थ्री-फेस जनरेटर आउटपुट

PLC कनेक्टशन में 3ϕ पावर सप्लाई की आवश्यकता होती है जिसे 3ϕ जनरेटर आउटपुट स्क्रीन की सहायता से देखा जा सकता है। जनरेटर तीन प्रकार की वाइंडिंग से फेस A, फेस B तथा फेस C से जुड़ा रहता है। इसमें चौथा कनेक्शन न्यूट्रल से कनेक्ट रहता है। यह 3ϕ वोल्टेज सोर्स के माध्यम से 120 डिग्री पर एक-दूसरे से कनेक्ट रहते हैं। इसमें तीनों फेस एक निश्चित क्रम में जुड़े रहते हैं तथा जिनमें 120/208 V AC



सप्लाई दी जाती है।

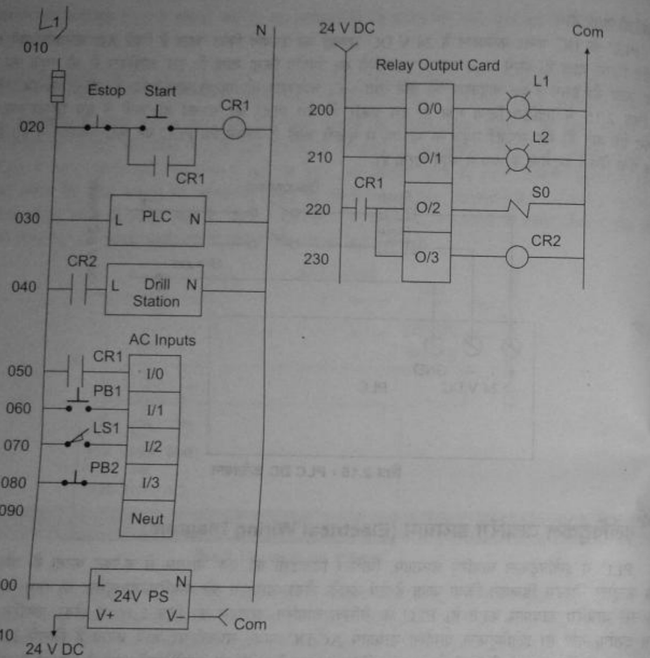
PLC के DC पावर कनेक्शन में 24 V DC सप्लाई का उपयोग किया जाता है जिसे AC कनेक्शन की तरह ही ऑपरेट किया जाता है। इसमें + ve तथा – ve वायरों का उपयोग किया जाता है। इस ऑपरेशन से भी फ्यूज का उपयोग किया जाता है। इसमें + ve कनेक्शन को हॉट तथा – ve कनेक्शन को न्यूट्रल कहा जाता है। PLC के DC कनेक्शन को चित्र 2.15 में प्रदर्शित किया गया है। इस प्रकार से जब PLC को सप्लाई दी जाती है तब डिसकनेक्ट सर्किट कनेक्ट हो जाते हैं। यह सप्लाई फ्यूज के माध्यम से गुजारी जाती है जिससे कि PLC को कोई नुकसान न हो। PLC में फ्यूज एक सेफ्टी इन्ट्रूमेन्ट के रूप में कार्य करता है।




चित्र 2.15 : PLC DC कनेक्शन

## इलेक्ट्रिकल वायरिंग डायग्राम (Electrical Wiring Diagram)

PLC में इलेक्ट्रिकल वायरिंग डायग्राम, विभिन्न डिवाइसों को एक माध्यम से कनेक्ट करता है। इस प्रकार जब एक कन्ट्रोल सिस्टम डिजाइन किया जाता है एवं उसके लैडर डायग्राम को वायरिंग डाक्यूमेन्ट के साथ बनाया जाता है तब उसे वायरिंग डायग्राम कहते हैं। PLC के बेसिक वायरिंग डायग्राम को चित्र 2.16 में लैडर वायरिंग डायग्राम के साथ दर्शाया गया है। इलेक्ट्रिकल वायरिंग डायग्राम AC/DC पावर सप्लाई पर कार्य करता है जिसमें 120 V AC या 220 V AC पावर सप्लाई दी जाती है। इस वायरिंग डायग्राम में कनेक्टिंग लाइनें पायी जाती हैं जिनकी नम्बरिंग की जाती है। इस लाइनों के नम्बरों से इलेक्ट्रिकल वायर कनेक्ट रहते हैं। लाइन 010 का उपयोग सिस्टम में पावर एन्ट्री के लिए किया जाता है। इस डायग्राम में फ्यूज का उपयोग सिस्टम में मैक्सिमम करंट ड्रॉप होने के लिए किया जाता है। लाइन 020 का उपयोग सिस्टम में आउटपुट कन्ट्रोल पावर के लिए किया जाता है। इस सिस्टम में स्टॉप बटन जब नॉर्मली क्लोज्ड होता है तब स्टार्ट बटन नॉर्मली ओपन होता है। इसमें ब्रांच तथा आउटपुट रंग (rung) CRI लगा होता है जो एक मास्टर कन्ट्रोल रिले होता है। PLC के 030 लाइन पर पावर रिसीव होती है। PLC के सभी इनपुट AC में होते हैं जो लाइन 050 से 090 तक स्थित रहते हैं। इसमें इनपुट/आउटपुट को मास्टर कन्ट्रोल रिले पर अवयवों के रूप में सेट किया जाता है तथा अन्य इनपुट नॉर्मली ओपन पुश बटन, लिमिट स्विच तथा नॉर्मली क्लोज पुश बटन क्रमशः लाइन 060, 070 तथा 080 उपस्थित रहते हैं। DC पावर सप्लाई को लाइन 100 पर दर्शाया गया है जो 24 V DC आउटपुट का उपयोग करती है। यह आउटपुट रिले पावर PLC के द्वारा ग्रीन इन्डीकेटर लाइट, रेड इन्डीकेटर लाइट, सोलेनॉइड तथा अन्य रिले को लाइन 200, 210, 220 तथा 230 के द्वारा कन्ट्रोल करती है। 230 लाइन पर रिले स्विचेज का एक सेट होता है जो ड्रिल सेक्शन को टर्न ऑन करता है।




चित्र 2.16 : इलेक्ट्रिकल वायरिंग डायग्राम

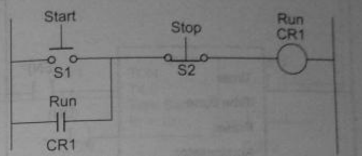
वायरिंग डायग्राम में नॉर्मली क्लोज स्टॉप बटन तथा नॉर्मली ओपन स्टार्ट बटन इच्छानुसार जोड़े जाते हैं जो वायरिंग डायग्राम की लाइन 220 में कन्सीडर किये गये हैं। इसमें यदि स्टॉप बटन को पुश किया जाता है तब यह स्विच को ओपन करता है तथा इससे पावर कन्ट्रोल रिले प्रभावित नहीं होती है। इस प्रकार आउटपुट पावर शटऑफ हो जाती है। यदि स्टॉप बटन डेमेज हो जाती है या वायर सिस्टम से अलग हो जाता है तब सिस्टम सेफ्टीपूर्वक शटडाउन हो जाता है। यदि पुश बटन नॉर्मली ओपन होती है तथा सिस्टम लगातार ऑपरेट होता है तब स्टॉप बटन पावर शटडाउन के लिए अयोग्य होती है। इस प्रकार से स्टार्ट बटन को कन्सीडर करते हैं। यदि बटन डेमेज है या वायर डिसकनेक्ट है तब यह सिस्टम को स्टार्ट करने में अयोग्य होता है।

## लैचेज (Latches)

लैचेज, PLC में एक रिले स्विय की तरह कार्य करता है। कभी-कभी रिले लैचेज को ऑन करके रखती है इसलिए यह डिवाइस एक्टिवेट होती है तब रिले स्विच ऑफ हो जाती है तथा लैचेज डिवाइस ऑन रहती है। लैचेज



साधारणतया क्षणिक पुश बटन बनाने में उपयोगी होता है जो एक स्विच को मेनटेन करके रखता है। उदाहरण के लिए, पुश बटन एक स्विच की तरह कार्य करता है जो किसी मशीन को ऑन तथा ऑफ कर सकता है। यह एक क्षणिक पुश बटन होता है जो एक सर्किट में वायर की सहायता से लैडर डायग्राम बनाने में सहायक होता है इसके सर्किट में ट्रांसफॉर्मर तथा फ्यूज उपयोग नहीं किये जाते हैं।




चित्र 2.17 : लैचेज सर्किट

जब सर्किट में पावर एप्लाई की जाती है तब प्रारम्भ में CR1 डीइनर्जाइज्ड तथा नॉर्मली ओपन रहता है। इसके पैरेलल में लगा S1 स्विच ओपन होता है। इस प्रकार से जब तक S1 स्विच को नहीं दबाया जाता है तब तक सम्पूर्ण सर्किट में किसी भी प्रकार की करंट प्रवाहित नहीं होती है। जब S1 स्विच को दबाया जाता है तब S1, S2 तथा CR1 के माध्यम से एक पाथ का निर्माण होता है जो CR1 को सक्रिय करता है। जैसे ही CR1 सक्रिय होता है वैसे ही CR1 के पैरेलल में लगा स्विच S1 क्लोज्ड हो जाता है। जब रिले कॉन्टेक्ट क्लोज्ड होता है तब स्विच S1 के माध्यम से वर्तमान प्रवाह के लिए एक पाथ बनाया जाता है जो नॉमली ओपन CR1 कॉन्टेक्ट तथा नॉर्मली क्लोज्ड S2 पुश बटन के कारण सम्भव होता है। इस बिन्दु पर स्विच S1 तथा रिले CR1 सक्रिय रहते हैं। नॉर्मली ओपन CR1 कॉन्टेक्ट सील या लैचेज की सहायता से सर्किट को ऑन करके रखते हैं। इस प्रकार के कॉन्टेक्ट में विभिन्न कॉन्टेक्ट विन्यास सीलिंग कॉन्टेक्ट, सील इन कॉन्टेक्ट तथा लैचिंग कॉन्टेक्ट शामिल किये जाते हैं।

स्टॉप स्विच S2 दबाने से सर्किट असक्रिय हो जाता है। इस इन्ट्रप्ट में करंट का फ्लो रंग (rung) से होता है और क्वॉइल CR1 असक्रिय रहती है तथा CR1 कॉन्टेक्ट स्विच S1 के साथ समान्तर में ओपन रहता है। जब S2 स्विच पुश रहता है तब रंग में किसी भी प्रकार की करंट प्रवाहित नहीं होती है क्योंकि S1 और CR1 दोनों नॉर्मली ओपन कॉन्टेक्ट ओपन रहते हैं।

लैचेज सर्किट की एक अन्य विशेषता यह है कि यह किसी एक स्विच को लगातार उपयोग में नहीं लाता है। जब पावर फेल होती है तब मशीन ऑन रहती है। इस प्रकार से लैच रंग असक्रिय रहता है। जब पावर रीस्टोर की जाती है तब मशीन पुन: स्टार्ट नहीं होती है। यह मैनुअल स्विच S1 दबाने से स्टार्ट की जाती है। इस स्विच की सहायता से किसी भारी मशीन को भविष्य में सुरक्षित रखा जा सकता है।

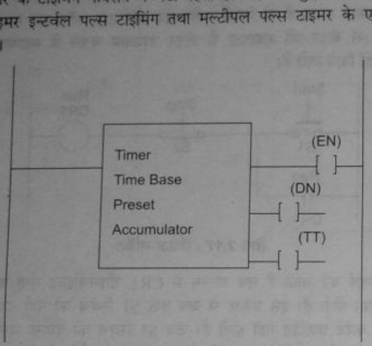
## PLC टाइमर (PLC Timer)

PLC टाइमर का फॉर्मेट चित्र 2.18 में दर्शाया गया है।

टाइमर फॉर्मेट में तीन आउटपुट इनेबल बिट (EN), डन बिट (DN) तथा टाइम टाइमिंग बिट (TT) होती हैं। इसमें प्रीसेट वेल्यू के द्वारा किया कन्ट्रोलर के टाइमर को पुन: सेट किया जाता है तथा एक्यूमुलेटर का उपयोग किसी डिवाइस के टाइम को लगातार अपडेट करने में किया जाता है।

PLC टाइमर, प्रोसेस कन्ट्रोल में सबसे अधिक उपयोग किया जाने वाला टाइमर है। अधिकतर कॉमन टाइमिंग फंक्शन डिले ऑन टाइम पर आधारित होते हैं जो किसी ऑपरेशन में एक बेसिक फंक्शन है। यह विभिन्न प्रकार के टाइमिंग कॉन्फिग्युरेशन का फंक्शन है जिसमें सभी फंक्शन एक या एक से अधिक टाइम डिले ऑन फंक्शन में बंटे रहते हैं। PLC विभिन्न प्रकार के टाइमिंग फंक्शन गुणकों को रखने की क्षमता रखता है। साधारण PLC टाइम डिले ऑन

फंक्शन तथा अन्य सात प्रकार के टाइमिंग फंक्शन में बँटा रहता है। यह बँटे हुए फंक्शन्स टाइम डिले ऑफ में भी के होते हैं। यह ऑफ डिले टाइमर इन्टर्वल पल्स टाइमिंग तथा मल्टीपल पल्स टाइमर के एक या एक से अधिक प्रोसे ऑपरेशन को सम्पन्न करते हैं।




चित्र 2.18 : PLC टाइमर का फॉर्मेट

अधिकतर PLCs में एक टाइमर फंक्शन होता है जिसे रिटेन्टिव टाइम डिले ऑन फंक्शन से जाना जाता है। कु PLCs एडीशनल में टाइम ऑन तथा टाइम ऑफ डिले फंक्शन भी रखते हैं। सामान्यत: PLC में दो प्रकार के बेसि PLC फंक्शनल ब्लॉक होते हैं। टाइमिंग ब्लॉक फंक्शन्स का उपयोग विभिन्न प्रकार के कॉन्टेक्ट अरेन्जमेन्ट तथा टाइमि टास्क में किया जाता है। इन्डस्ट्रियल टाइमिंग टास्क में बेल्डिंग, पेन्टिंग तथा हीट ट्रीटमेंट आदि को शामिल किया जात है। इस प्रकार से टाइमर दो ऑपरेशनों के बीच का समय निर्धारण करता है।

PLC टाइमर फंक्शन्स के द्वारा किसी भी इन्डस्ट्रियल टाइमर को बदलता जा सकता है यद्यपि इन्डस्ट्रियल टाइम मोटर द्वारा चलित होना चाहिए। इन्डस्ट्रियल टाइमर PLC के माध्यम से किसी भी ऑपरेशन को एक निश्चि समयान्तराल में सम्पन्न करता है जिसमें एक निश्चित PLC का उपयोग किया जाता है। अत: इस प्रकार यह कहा ज सकता है कि टाइमर में उपयोग किया जाने वाला समय किसी प्रोसेस के लिए एक महत्वपूर्ण समय होता है जिसे टाइम डिवाइस द्वारा निर्धारित किया जाता है।

**उदाहरण**—दो मोटर के लिए लैडर डायग्राम बनाइए जो निम्नलिखित शर्तें रखती है—

(i) पुश स्टार्ट बटन दबाने पर मोटर 1 स्टार्ट हो,

(ii) 10 सेकण्ड बाद मोटर 2 ऑन हो,

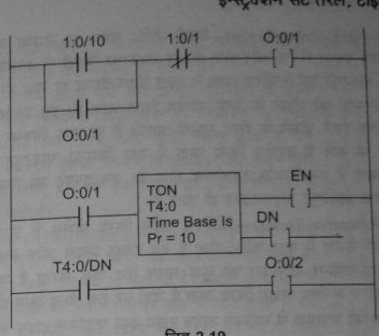
(iii) स्टॉप बटन दबाने पर मोटर 1 और 2 ऑफ हो।

दिया है—बेस टाइम 1 सेकण्ड

| इनपुट | आउटपुट |
|---|---|
| स्टार्ट = इनपुट = 0 / 10 | मोटर 1 = आउटपुट : 0/1 |
| स्टॉप = इनपुट = 0 / 1 | मोटर 2 = आउटपुट : 0/2 |

इस प्रकार से इनपुट तथा आउटपुट की सहातया से चित्र 2.19 में लैडर डायग्राम दर्शाया गया है जो निम्नलिखि है—






चित्र 2.19

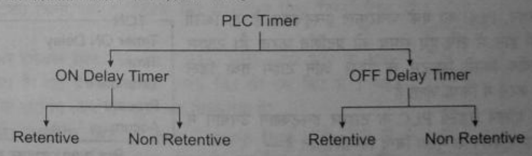
## टाइमर का क्लासीफिकेशन (Classification of Timers)

### टाइमर (Timer)

टाइमर एक डिवाइस है जो टाइम डिले की स्थिति को सर्किट या सिस्टम की सहायता से ऑन तथा ऑफ के रूप में प्रदर्शित करता है। टाइमर मुख्य रूप से इलेक्ट्रोमैकेनिकल टाइमर, इलेक्ट्रॉनिक टाइमर तथा PLC टाइमर पर आधारित होता है।

### PLC टाइमर का क्लासीफिकेशन (Classification of PLC Timers)

PLC टाइमर का क्लासीफिकेशन नीचे दर्शाया गया है—




PLC टाइमर का क्लासीफिकेशन निम्नलिखित प्रकार से है—

**(i) ऑन डिले टाइमर** (ON Delay Timer)—ऑन डिले टाइमर में जब इनपुट स्विच ऑन होता है तब आउटपुट प्रीसेट टाइम इनर्जाइज्ड होता है। जब इनपुट स्विच ऑफ होता है तब आउटपुट तुरन्त ही डीइनर्जाइज्ड हो जाता है। ऑन डिले टाइमर का उपयोग मुख्य रूप से टाइमिंग फंक्शन में किया जाता है।

उदाहरण के लिए, एक ओवन को खाना पकाने के लिए इस्तेमाल किया गया है। खाना पकने का टाइम 10 मिनट है। यदि पकने वाले खाने को हीटर शुरू करने के 30 सेकण्ड बाद ओवन में रखा जाता है जो यह निश्चित करें कि इसे मैनुअल हस्तक्षेप करने की आवश्यकता तो नहीं है। इस स्थिति को ऑन डिले टाइम द्वारा संभाला जाता है व खाने को ओवन में रखा जाता है तब ऑन डिले टाइमर स्विच ऑन करता है। जब ओवन स्विच ऑन होता है तब टाइमर हीट करने के लिए 30 सेकण्ड का इंतजार करता है। टाइमर इनपुट के स्विच ऑफ होने के लिए 10 मिनट काउन्ट करता है। जब इनपुट स्विच ऑफ होता है तब हीटर तुरन्त ही डीइनर्जाइज्ड हो जाता है।

**(ii) ऑफ डिले टाइमर** (OFF Delay Timer)—किसी प्रीसेट समय पर उसका आउटपुट ऑफ होता है, इ ऑफ डिले टाइमर कहते हैं। जब इनपुट स्विच ऑन होता है तब आउटपुट जल्दी से इनर्जाइज्ड हो जाता है। यद्यपि ज इनपुट स्विच ऑफ होता है तब आउटपुट पूर्व निर्धारित टाइम से पहले डीइनर्जाइज्ड हो जाता है।

उदाहरण के लिए, एक मिक्सर को पीसने के लिए उपयोग किया जाता है। इस प्रकार से किसी सामग्री को पीस के लिए पाँच मिनट लगते हैं तब इसमें पीसने के लिए सामग्री डालते हैं जो एक मिक्सर कन्ट्रोल का कार्य करती इसमें किसी सामग्री को इनपुट के रूप में उपयोग किया जाता है तथा जिसका आउटपुट मिक्सर होता है। इसमें ज इनपुट को स्विच ऑन किया जाता है तब मिक्सर इनर्जाइज्ड होता है। जब इनपुट को स्विच ऑफ किया जाता है त मिक्सर चलना बंद हो जाता है तथा आउटपुट डीइनर्जाइज्ड हो जाता है।

**(iii) रिटेन्टिव टाइमर** (Retentive Timer)—रिटेन्टिव टाइम किसी प्रोसेस में लगने वाले समय पर आधारि होता है। इस प्रकार से जब किसी प्रोसेस में कुछ फॉल्ट होता है तब इनपुट स्विच ऑफ होता है। जब इनपुट पुनः स्वि ऑन किया जाता है तब यह उस टाइमिगं ऑपरेशन को कुछ समय बाद शुरू करता है जहाँ पर फॉल्ट हुआ था। इ प्रकार के टाइमर को उस समयान्तराल के लिए प्रोग्राम किया जाता है जहाँ पर ट्रेक वेल्यू खत्म होती है।

इस प्रकार से रिटेन्टिव टाइम की सहायता से मशीनरी डाउन टाइम तथा साइकिल टाइम को मेजर किया जाता है।

उदाहरण के लिए, पैकेजिंग प्लांट में, 100 कार्टन के पैकिंग में लगे हुए समय को रिटेन्टिव टाइमर द्वारा काउन्ट किया जाता है।

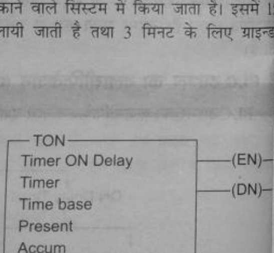
**(iv) नॉन-रिटेन्टिव टाइमर** (Non Retentive Timer)—नॉन रिटेन्टिव टाइमर वह टाइमर है जो किसी ऑपरेश के दौरान फॉल्ट के कारण उपयोग में नहीं लाया जाता है। जब किसी प्रोसेसर में फॉल्ट होता है तब इनपुट स्विच ऑफ होता है। इस प्रकार से जब इनपुट को पुनः ऑन किया जाता है तब इसके द्वारा नये समय की गिनती की जाती है।

उदाहरण के लिए, नॉन-रिटेन्टिव टाइमर का उपयोग किसी खाना पकाने वाले सिस्टम में किया जाता है। इसमें 1 मिनट का टाइम सेट किया जाता है। इसमें 2 मिनट के लिए मोटर चलायी जाती है तथा 3 मिनट के लिए ग्राइन्ड चलाया जाता है।

## टाइमर इन्स्ट्रक्शन (Timer Instruction)

टाइमर इन्स्ट्रक्शन, PLC का एक पावरफुल इन्स्ट्रक्शन है जो किसी ऑपरेशन के परफॉर्म होने में लगे हुए समय को प्रदर्शित करता है। टाइमर इन्स्ट्रक्शनों का उपयोग किसी सिस्टम के डिले ऑन टाइम तथा डिले ऑफ टाइम को ज्ञात करने में किया जाता है

चित्र 2.20 मे एलिन ब्रेडली PLC के टाइमर इन्स्ट्रक्शन उपयोग में लाये गये हैं। टाइमर इन्स्ट्रक्शन के मुख्य बिन्दु निम्नलिखित हैं—



चित्र 2.20 : टाइमर इन्स्ट्रक्शन

**(i) पैरामीटर्स** (Parameters)—टाइमर इन्स्ट्रक्शन पैरामीटर ब्लॉक फॉर्मेट में प्रदर्शित किये जाते हैं तथा उन्हें एक निश्चित टाइम के लिए उपयोग किया जाता है। पैरामीटर में उपयोग कि जाने वाले प्रमुख टाइमर इन्स्ट्रक्शन निम्नलिखित हैं—

**( अ ) टाइमर** (Timer)—इसका उपयोग डाटा स्टोरेज फाइल में टाइमर एड्रेस को स्पेसीफाइ करने के लिए किया जाता है। टाइमर एड्रेस का फॉर्मेट TX : Y होता है जिसमें T फाइल टाइप रिप्रेजेन्ट करता है जो एक टाइमर होता है इस स्थिति में डिफॉल्ट फाइल नम्बर को 4 से प्रदर्शित किया जाता है। इसमें X टाइमर फाइल नम्बर को प्रदर्शित करता है। इसकी वेल्यू को 3 से 999 तक बढ़ाया जा सकता है। इसमें Y टाइमर नम्बर को प्रदर्शित करता है। इसकी वेल्यू को 0 से 999 तक बढ़ाया जा सकता है।

**( ब ) टाइमर बेस** (Timer Base)—यह टाइम यूनिट को स्पेसीफाइ करता है जिसमें समय परिवर्तन के अनुसार एड्रेस को बदला जाता है। इसमें मुख्य रूप से 1, 0.1 तथा 0.01 सेकण्ड टाइम यूनिट का उपयोग किया जाता है।



**( स ) प्रीसेट** (Preset)—यह बढ़ते हुए समय के द्वारा स्पेसीफाइ किया जाता है जिसमें समय को काउन्ट किया जाता है। इसकी वेल्यू को 0 से 32767 तक बढ़ाया जा सकता है।

**( द ) वेल्यू** (Value)—इसमें टाइम बेस तथा प्रीसेट पैरामीटर के द्वारा लिए गये समय को टाइमर द्वारा निर्धारित किया जाता है। उदाहरण के लिए, यदि टाइम बेस की वेल्यू 0.01 सेकण्ड है तथा प्रीसेट वेल्यू 450 है तब टाइमर 4.5 सेकण्ड में एक निश्चित क्रिया सम्पन्न करता है। यदि टाइम बेस वेल्यू 1 सेकण्ड है तथा प्रीसेट वेल्यू 450 है तब टाइमर द्वारा 450 सेकण्ड में एक निश्चित क्रिया सम्पन्न की जायेगी।

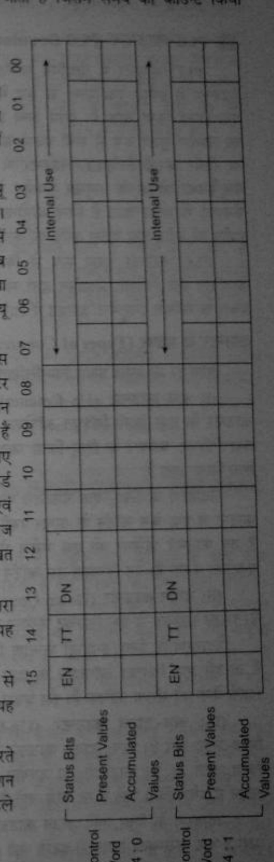
**( इ ) एक्यूमुलेटेड वेल्यू** (Accumulated Value)—एक्यूमुलेटेड वेल्यू के माध्यम से किसी प्रोसेस में कितना टाइम लगा, यह काउन्ट किया जाता है। इसकी वेल्यू को 0 से 32767 तक बढ़ाया जा सकता है जबकि प्रोग्रामिंग में इसकी वेल्यू को 0 सेट किया जाता है। इसमें प्रोसेसर जब अपडेट होता है तब प्रोग्राम रन होता है। उदाहरण के लिए, यदि टाइम बेस 0.1 सेकण्ड है तथा एक्यूमुलेटेड की स्पेसीफाइड वेल्यू 10 है तब एक्यूमुलेटेड पैरामीटर की वेल्यू 2 सेकण्ड बाद 30 होगी।

**(ii) स्टेटस बिट्स** (Status Bits)—स्टेटस बिट्स आउटपुट इन्स्ट्रक्शन्स है जो ब्लॉक फॉर्मेट का अनुशरण करता है। इसमें प्रोसेसर के द्वारा पैरामीटर की वेल्यू को 48-बिट स्टेक्चर के रूप में स्टोर करता है, जिसको तीन सेक्शन में बाँटा जा सकता है। इसके प्रत्येक टाइमर एरिया में 16-बिट वर्ड होते हैं जिन्हें T से प्रदर्शित किया जाता है। प्रथम वर्ड का उपयोग प्रोग्रामिंग के लिए किया जाता है जिसे प्रोसेसर आन्तरिक रूप से उपयोग में लाता है। द्वितीय वर्ड का उपयोग प्रीसेट वेल्यू के पैरामीटर को स्टोर करने मे किया जाता है एवं तीसरे वर्ड एक्यूमुलेटर वेल्यू को स्टोर करता है। चित्र 2.21 मे टाइमर स्टोरेज के फॉर्मेट को प्रदर्शित किया गया है। स्टेटस बिट के प्रमुख बिन्दु निम्नलिखित हैं—

**( अ ) टाइमर इनेबल बिट** (Timer Enable Bit)—इसे EN के द्वारा प्रदर्शित किया जाता है। यह प्रथम वर्ड की 15th बिट है। इस बिट मे 1 यह प्रदर्शित करता है कि टाइमिंग ऑपरेशन शुरू कर दिया गया है।

**( ब ) टाइमर टाइमिंग बिट** (Timer Timing Bit)—इसे TT से प्रदर्शित करते हैं। यह प्रथम वर्ड की 14th बिट है। इस बिट में वेल्यू 1 यह प्रदर्शित करती है कि टाइमिंग ऑपरेशन ऑन है।

**( स ) टाइम डन बिट** (Time Done Bit)—इसे TN से प्रदर्शित करते हैं। यह प्रथम वर्ड की 13th बिट है। इस बिट में 1 वेल्यू टाइमिंग ऑपरेशन को कम्प्लीट प्रदर्शित करती है। अतः इस प्रकार से ऑन डिले तथा ऑफ डिले टाइमर ऑपरेशन को TT तथा DN बिट से कन्ट्रोल किया जाता है।



चित्र 2.21 : टाइमर स्टोरेज फॉर्मेट

## PLC में उपयोग होने वाले काउन्टर (Counter Used in PLC)

### PLC काउन्टर (PLC Counter)

PLC काउन्टर में प्रोग्रामिंग फॉर्मेट पाया जाता है जो टाइमर फॉर्मेट के समान ही होता है। इस प्रकार से PL काउन्टर में इनपुट एक पल्स के रूप में दिया जाता है जो किसी वेरियेबल को काउन्ट करता है जिसे PLC के द्वा एनालाइज्ड किया जाता है। इसमें अन्य इनपुटों को इनेबल/रीसेट के रूप में उपयोग किया जाता है। रिवॉल्यूशन काउ का उपयोग मुख्य रूप से सभी प्रकार की मैन्यूफैक्चर इन्डस्ट्रीज में किया जाता है जो किसी प्रोडक्ट या मशीन के घू पर निर्भर करती है। PLC काउन्टर में इनेबल तथा रीसेट के लिए अलग-अलग इनपुट प्रदान किये जाते हैं जि मैन्यूफैक्चर प्रोसेस के अनुसार उपयोग में लाया जाता है। PLC काउन्टर को मैकेनिकल, इलेक्ट्रिकल तथा इलेक्ट्रॉनि फंक्शनों से जोड़ा जाता है जिन्हें कन्वेन्शनल काउन्टर के माध्यम से रिप्लेस किया जाता है। काउन्टर का उपयोग कि मशीन की स्पीड तथा पार्टस आदि को काउन्ट करने में किया जाता है।

PLC काउन्टर मुख्य रूप से किसी इन्डस्ट्रीज में प्रोडक्ट तथा पैकेट या कार्टन को काउन्ट करता है। इस काउन्टिंग एक निश्चित प्रोग्रामिंग द्वारा सम्पन्न की जाती है जिसमें विभिन्न प्रकार के सेन्सर, सोलेनॉइड वाल्व तथा अ प्रकार के सेन्सिंग इन्स्ट्रूमेन्ट उपयोग मे लाये जाते हैं।

**काउन्टर के प्रकार (Types of Counter)**

काउन्टर के प्रमुख प्रकार निम्नलिखित हैं—

**(i) अप काउन्टर** (Up Counter)—अप काउन्टर का उपयोग अधिकतर ऑपरेशनों में किया जाता है। अ काउन्टर के द्वारा किसी निश्चित ऑपरेशन को काउन्ट किया जाता है। काउन्टर ऑपरेशन का उपयोग एक निश्चित स्थ तथा निश्चित फंक्शन के लिए किया जाता है। इस काउन्टर का उपयोग एक से अधिक वस्तुओं को काउन्ट करने लिए किया जाता है।

उदाहरण के लिए, एक पैकेजिंग प्लांट में किसी कॉर्टन को 12 केन के द्वारा भरा जाता है। इस प्रकार से प्रत्ये काउन्ट में एक केन कॉर्टन के अन्दर रखी जाती है जिसमें एक नम्बर की वृद्धि होती है। जब केन कॉर्टन में पहुँच ज हैं तब काउन्टर प्रोसेसर को एक मैसेज भेजता है कि कॉर्टन फुल हो गया है। इस प्रकार से प्रोसेसर अन्य डिवाइस एक्टीवेट करता है। यह डिवाइस उस कॉर्टन को सील करती है तथा उसे असेम्बली लाइन की ओर मूव कराती है।

**(ii) डाउन काउन्टर** (Down Counter)—डाउन काउन्टर का उपयोग भी अप काउन्टर की तरह ही किया ज है। इसकी सहायता से एक निश्चित फंक्शन को परफॉर्म किया जाता है। इसमें काउन्ट नम्बर को घटाया जाता है।

उदाहरण के लिए, इनवेन्ट्री कन्ट्रोल सिस्टम के जूस केन पैकेजिंग प्लांट में यदि इसमें खाली केन की मात्रा अधि है तो इसे एक निश्चित ऑपरेशन के द्वारा डाउन काउन्ट किया जाता है तथा काउन्टर प्रोसेसर को मैसेज भेजता है खाली केनों की संख्या अधिक है। इस प्रकार से इन केनों को पुन: भरने के लिए असेम्बली लाइन पर भेजा जाता है।

**(iii) अप-डाउन काउन्टर** (Up-Down Counter)—अप-डाउन काउन्टर अप तथा डाउन काउन्टर मिला-जुला रूप है। प्राय: डाउन काउन्टर का उपयोग अकेले नहीं किया जाता है। इसका उपयोग अप काउन्टर के स किया जाता है। इन काउन्टरो का उपयोग मुख्यत: इनवेन्ट्री कन्ट्रोल में किया जाता है जिसमें प्रोडक्ट या तो असेम्ब लाइन में प्रवेश किये जाते हैं या छोड़े जाते हैं। इन काउन्टरों को ट्रेक डिफेक्टर पार्ट्स भी कहा जाता है।

उदाहरण के लिए, यदि डाउन काउन्टर के द्वारा 10 पार्ट असेम्बली लाइन पर दिखाये जाते हैं तब अप काउन्टर द्वारा 8 पार्ट असेम्बली लाइन से हटाये जाते हैं जिसमें 2 डिफेक्टिव पार्ट होते हैं।

**अप, डाउन तथा अप-डाउन काउन्टर में अन्तर (Difference Between Up, Down and Up-Down Counters)**—अप, डाउन तथा अप-डाउन काउन्टर में प्रमुख अन्तर निम्नलिखित हैं—



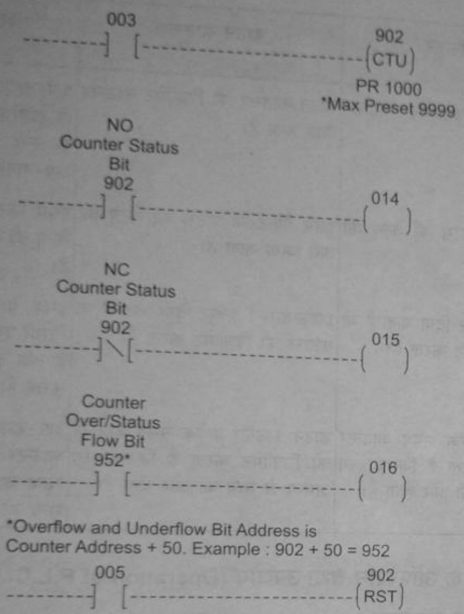
| क्र० सं० | अप काउन्टर | डाउन काउन्टर | अप-डाउन काउन्टर |
|---|---|---|---|
| (i) | अप काउन्टर को इन्क्रीमेंट काउन्टर कहा जाता है। | डाउन काउन्टर को डिक्रीमेंट काउन्टर कहा जाता है। | अप-डाउन काउन्टर के द्वारा स्थिति या तो डायरेक्शन के रूप में या कन्ट्रोल के रूप में बदली जाती है। इसे अप-डाउन काउन्टर कहते हैं। |
| (ii) | इसमें पैकेट या पार्ट्स की वेल्यू को बढ़ाया जाता है। | इसमें पैकेट या पार्ट्स की वेल्यू को कम किया जाता है। | इसमें किसी भी पैकेट या पार्ट्स की वेल्यू को कम तथा अधिक किया जाता है। |
| (iii) | इसके द्वारा 1 इनुपट दिया जाता है जो काउन्ट को इन्क्रीमेंट करता है। | इसके द्वारा 1 इनपुट दिया जाता है जो काउन्ट को डिक्रीमेंट करता है। | इसके द्वारा 2 इनपुट दिये जाते हैं, जिसमें एक इनपुट को इन्क्रीमेंट करता है तथा दूसरा काउन्ट को डिक्रीमेन्ट करता है। |
| (iv) | अप काउन्टर प्रत्येक समय काउन्टर को इन्क्रीमेन्ट करता है जिसका रंग असत्य से सत्य की ओर होता है। | डाउन काउन्टर प्रत्येक समय काउन्टर को डिक्रीमेंट करता है जिसका रंग असत्य से सत्य की ओर होता है। | अप-डाउन काउन्टर में प्रत्येक समय काउन्टर को इन्क्रीमेंट तथा डिक्रीमेंट किया जाता है जिसका रंग असत्य से सत्य की ओर होता है। |

## PLC काउन्टर के ऑपरेशन तथा उपयोग (Operation of P.L.C. Counter and Uses)

### PLC काउन्टर का ऑपरेशन (Operation of PLC Counter)

PLC में काउन्टर इन्स्ट्रक्शन, टाइमर इन्स्ट्रक्शन की तरह ही ऑपरेट होते हैं। टाइमर तथा काउन्टर के बीच बड़ा अन्तर यह है कि टाइमर इन्स्ट्रक्शन एक्यूमुलेटिड वेल्यू को लगातार बढ़ाता रहता है तब इनेबल कॉन्टेक्ट ऑन होता है। अन्य प्रकार से, काउन्टर सम्पूर्ण कॉन्टेक्ट ट्रांजीशन को एक्यूमुलेटिड वेल्यू को 0 तथा 1 के फॉर्म में बढ़ाता है। इसका मतलब यह है कि कॉन्टेक्ट अपनी पुन: स्थिति में 0 के रूप में आ जाता है जिससे यह ट्रांजीशन को दोबारा रख सकता है। काउन्टर में कॉन्टेक्ट के ट्रांजीशन को किसी भी समय स्टे किया जा सकता है। इसमें ट्रांजीशन के स्थान पर क्या महत्वपूर्ण है उसे काउन्टर में उपयोग किया जाता है। चित्र 2.22 में काउन्टर के इन्स्ट्रक्शनो को प्रीसेट ऑपरेशन के साथ दर्शाया गया है।

काउन्टर में टाइमर की तरह प्रीसेट तथा एक्यूमुलेटिव वेल्यू पायी जाती है। प्रीसेट वेल्यू काउन्टर के लिए डिजायर्ड वेल्यू है जिसको इच्छानुसार लम्बी वेल्यू को काउन्ट करने में उपयोग किया जाता है। काउन्टर के लिए मैक्सिमम प्रीसेट वेल्यू 9999 है। काउन्टर इस वेल्यू के बीच में अनेक कन्ट्रोल ऑपरेशन परफॉर्म करता है। जब एक्यूमुलेटर वेल्यू प्रीसेट वेल्यू के बराबर होती है तब इसकी स्टेटस बिट उच्च होती है। इसकी स्टेटस बिट सभी नॉर्मली ओपन तथा नॉर्मली क्लोज कॉन्टेक्ट को कन्ट्रोल करती है जिसमें काउन्टर के सभी नम्बर समान होते हैं। काउन्टर भी इसी प्रकार की बि को रखता है जो यह इंगित करता है कि काउन्टर ओवरफ्लो स्टेट में है या अन्डरफ्लो स्टेट में है। जब काउन्टर ओवरफ्लों स्टेट में होता है तब एक्युमुलेटर 9999 पर होता है तथा जब अन्डरफ्लो स्टेट में होता है तब यह 0 पर होता है। ओवरफ्लो तथा अन्डरफ्लो कन्डीशन समान बिट को काउन्टर में सेट करके रखती है। बिट एड्रेस को प्राप्त करने के लिए काउन्टर में 50 जोड़ा है। उदाहरण के लिए, यदि काउन्टर एड्रेस 902 है तथा ओवरफ्लो बिट के द्वारा किसी कॉन्टेक्ट को कन्ट्रोल किया जाता है तब इसका एड्रेस 952 होता है।




चित्र 2.22 : काउन्टर ऑपरेशन

काउन्टर में टाइमर की तरह प्रीसेट इन्स्ट्रक्शन होता है जो PR वेल्यू को काउन्टर में देने के लिए उपयोग कि है। जब अप काउन्टर इन्स्ट्रक्शन का उपयोग किया जाता है तब इसकी प्रीसेट वेल्यू शून्य से स्टार्ट होती है त डाउन काउन्टर इन्स्ट्रक्शन का उपयोग किया जाता है तब इसकी प्रीसेट वेल्यू इसके प्रीसेट वेल्यू तथा शून्य पर ख है। इस प्रकार से जब स्टेटस बिट को इनर्जाइज्ड किया जाता है तब इसकी प्रीसेट वेल्यू एक्यूमुलेटर वेल्यू के सम तथा इसकी एक्यूमुलेटर वेल्यू डाउन काउन्टर के लिए शून्य के बराबर होगी।

## PLC काउन्टर के उपयोग (Applications of PLC Counter)

PLC काउन्टर के प्रमुख उपयोग निम्नलिखित हैं—

(i) PLC काउन्टर का उपयोग अप/डाउन काउन्टर में किया जाता है जो विभिन्न प्रकार के ऑपरेशनों परफॉर्म करता है।



## PLC के काउन्टर पैरामीटर तथा स्टेटस बिट कन्ट्रोल (Counter Parameter of PLC and Status Bit Control)

### काउन्टर पैरामीटर (Counter Parameter)

पैरामीटर से किसी नम्बर की स्थिति को बढ़ाया या घटाया जाता है जिसे काउन्टर द्वारा एक निश्चित पाथ दिया जाता है। PLC काउन्टर में मुख्य रूप से दो पैरामीटर होते हैं जो निम्नलिखित हैं—

(i) **एक्यूमुलेटर पैरामीटर** (Accumulator Parameter)—एक्यूमुलेटर पैरामीटर में एक्यूमुलेटर वेल्यू असत्य से सत्य ट्रांजीशन में होती है। इसमें जब काउन्टर को किसी वेल्यू से प्रीसेट किया जाता है तब यह उत्पन्न होता है।

(ii) **प्रीसेट पैरामीटर** (Preset Parameter)—प्रीसेट पैरामीटर वेल्यू को PRE से प्रदर्शित किया जाता है। इसमें स्पेसीफाइड वेल्यू को काउन्टर पर पहुँचाया जाता है जिससे कन्ट्रोलर डन बिट को सेट करता है। जब एक्यूमुलेटर वेल्यू प्रीसेट वेल्यू के बराबर या बड़ी होती है तब यह स्टेटस बिट सेट होती है। इसका उपयोग आउटपुट डिवाइस को कन्ट्रोल करने में किया जाता है।

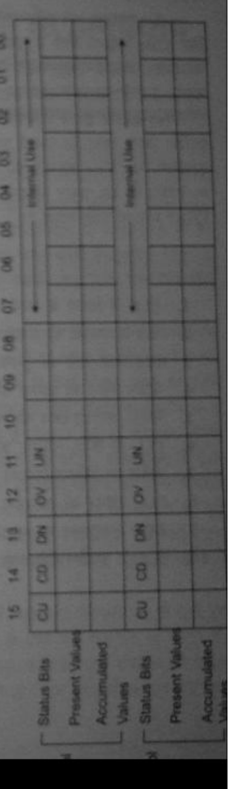
प्रीसेट तथा एक्यूमुलेटिड वेल्यू की काउन्टर रेन्ज – 32, 768 से + 32, 767 तक होती है जो साइन इन्टीजर में स्टोर होती है तथा ऋणात्मक वेल्यू 2's कॉम्प्लीमेन्ट के फॉर्म में स्टोर की जाती है।

### स्टेटस बिट कन्ट्रोल (Stutus Bit Control)

काउन्टर का स्टोरेज फॉर्मेट टाइमर के समान ही होता है। इसमें प्रोसेसर 48-बिट स्टेक्चर में विभिन्न प्रकार के पैरामीटर की वेल्यू को स्टोर करता है जो 16-बिट वर्ड रूप में काउन्टर एरिया C तथा डाटा मैमोरी में स्टोर होती है। इसके प्रथम वर्ड को प्रोसेसर द्वारा आन्तरिक प्रोग्रामिंग में उपयोग किया जाता है। इसमें द्वितीय वर्ड के द्वारा प्रीसेट पैरामीटर की वेल्यू को स्टोर किया जाता है तथा तीसरे वर्ड द्वारा एक्यूमुलेटर की वेल्यू को स्टोर किया जाता है। काउन्टर स्टोरेज फॉर्मेट को चित्र 2.23 में दर्शाया गया है।

इस प्रकार से जब प्रोसेसर स्टेटस बिट की वेल्यू को बदला जाता है तब काउन्टर इन्स्ट्रक्शन अनकाउन्टिड होता है। स्टेटस बिट मे उपयोग की जाने वाली प्रमुख बिटें निम्नलिखित हैं—

(i) **काउन्ट अप इनेबल बिट** (Count Up Enable Bit)—काउन्ट अप इनेबल बिट को CU से प्रदर्शित किया जाता है। यह प्रथम वर्ड के 15th बिट पर होती है। इसमें वेल्यू 1 अप काउन्टर के इन्स्ट्रक्शन को प्रारम्भ करती है। जब काउन्ट अप इनेबल बिट की वेल्यू 1 होती है तब रंग सत्य होता है तथा 0 होती है तब रंग असत्य होता है। इसमें RFS इन्स्ट्रक्शन का उपयोग कर रीसेट वेल्यू को शून्य किया जाता है।



जाता है।

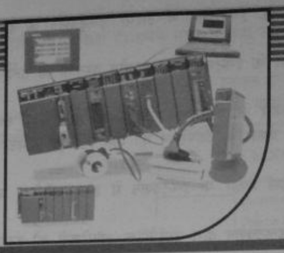
**(iii) काउन्ट अप/डाउन डन बिट** (Count Up/Down Done Bit)—इसे DN से प्रदर्शित किया जाता है। य बिट प्रथम वर्ड के 13th बिट पर होती है। इस बिट की वेल्यू जब 1 होती है तब यह काउन्टिंग ऑपरेशन को कम्प्ली दर्शाता है तथा एक्यूमुलेटिड वेल्यू प्रीसेट वेल्यू के बराबर या अधिक होती है। उदाहरण के लिए, यदि काउन्टर क प्रीसेट वेल्यू 24 है तब DN बिट 1 होती है। जब रंग 24 के बीच में होता है तब यह असत्य से सत्य ट्रांजीशन क प्रदर्शित करता है।

**(iv) काउन्ट अप ओवरफ्लो बिट** (Count Up Overflow Bit)—इसको OV से प्रदर्शित करते हैं। इसकी प्रथ बिट 12th-बिट पर होती है, इस बिट की वेल्यू जब 1 होती है तब यह काउन्टिग बिट को + 32767 की अपर लिमि को दर्शाता है तथा एक्यूमुलेटिड वेल्यू को भी अपर लिमिट + 32767 पर दर्शाता है। इस प्रकार से इसका वार्पड अराण डाटा –32768 पर प्राप्त होगा तब प्रोसेसर इसकी वेल्यू को पीछे की ओर स्टोर करेगा।

**(v) काउन्ट डाउन ओवरफ्लो बिट** (Count Down Overflow Bit)—काउन्ट डाउन ओवरफ्लो बिट को UN से प्रदर्शित किया जाता है। इसकी प्रथम बिट 11th-बिट वर्ड पर होती है। इस बिट की वेल्यू जब 1 होती है तब य काउन्टिंग बिट को – 32768 की लोअर लिमिट को दर्शाता है तथा एक्यूमुलेटिड वेल्यू को भी लोअर लिमिट –3276 पर दर्शाता है। इस प्रकार से इसका वार्पड अराण्ड डाटा + 32768 पर प्राप्त होता है तब प्रोसेसर इसकी वेल्यू को 32767 से 0 पर स्टोर करता है।

## अभ्यासीय प्रश्न

1. PLC में रिलेज के कार्य को समझाइए।
2. रिले के प्रकारों को समझाइए।
3. PLC तथा रिले की तुलना कीजिए।
4. प्रोग्राम स्कैन की प्रोसेस को समझाइए।
5. मैमोरी आर्गेनाइजेशन को समझाइए।
6. PLC के लैडर लॉजिक को समझाइए।
7. लैचेज पर संक्षिप्त टिप्पणी लिखिए।
8. PLC टाइमर को उदाहरण की सहायता से समझाइए।
9. टाइमर इन्स्ट्रक्शन को समझाइए।
10. PLC में उपयोग होने वाले विभिन्न प्रकार के काउन्टर को समझाइए।



# 3 प्रोग्रामिंग टेक्निक्स तथा पीएलसी के अनुप्रयोग (Programming Techniques and Applications of P.L.C.)

## PLC इन्स्ट्रक्शन (PLC Instruction)

PLC की प्रोग्रामिंग लैडर लॉजिक डायग्राम या माइक्रोकन्ट्रोलर/माइक्रोप्रोसेसर पर आधारित होती है। इसमें मुख्य रूप से उन्हीं इन्स्ट्रक्शनों का उपयोग किया जाता है, जिनका उपयोग माइक्रोकन्ट्रोलर/माइक्रोप्रोसेसर में किया जाता है। इस प्रकार से किसी इन्स्ट्रक्शन के द्वारा PLC को यह सूचना दी जाती है कि उसे कौन-सा कार्य करना है। प्रत्येक PLC के लिए इन्स्ट्रक्शनो की संख्या निश्चित होती है जिन्हें वह निष्पादित कर सकता है। अत: किसी PLC द्वारा निष्पादित किये जाने वाले समस्त इन्स्ट्रक्शनों के समूह को उस PLC का इन्स्ट्रक्शन सेट कहते हैं।

PLC के लिए लैडर डायग्राम प्रोग्रामिंग लैंग्वेज अधिक उपयोग में लायी जाती है। लैडर डायग्राम एक सिम्बोलिक इन्स्ट्रक्शन है जिसका उपयोग प्रोग्रामेबल लॉजिक कन्ट्रोलर प्रोग्राम को उत्पन्न करने में किया जाता है। लैडर इन्स्ट्रक्शन सिम्बल्स का उपयोग कन्ट्रोल लॉजिक को प्राप्त करने तथा उसे मैमोरी में स्टोर करने के लिए किया जाता है। इसमें विभिन्न प्रकार के इन्स्ट्रक्शन सेट होते हैं जो रिले टाइप लॉजिक, टाइमिंग तथा काउटिंग एवं मैथ ऑपरेशन परफॉर्म करते हैं। इन्स्ट्रक्शन प्रोग्रामेबल कन्ट्रोलर मॉड्यूल पर निर्भर रहते हैं। इस प्रकार से इसके इन्स्ट्रक्शन सेट में परिवर्तन करके अन्य ऑपरेशनों को परफॉर्म किया जा सकता है। इसमें एनालॉग कन्ट्रोल, डाटा मैनिपुलेशन, रिपोर्टिंग, कॉम्पलेक्स कन्ट्रोल लॉजिक आदि फंक्शनों का उपयोग एडीशनल फंक्शन के रूप में किया जाता है। लैडर डायग्राम प्रोग्राम का मुख्य फंक्शन कन्ट्रोल आउटपुट है जो इनपुट कन्डीशन पर आधारित होता है। इस कन्ट्रोल का उपयोग लैडर रंग के माध्यम से पूरा किया जाता है। इस प्रकार से एक लैडर लॉजिक रंग में इनपुट कन्डीशन का सेट होता है जिसे रिले कॉन्टेक्ट टाइप इन्स्ट्रक्शन से प्रदर्शित किया जाता है। इसमें रंग के अन्त में आउटपुट इन्स्ट्रक्शन का उपयोग किया जाता है जो रिले क्वॉइल सिम्बल को प्रदर्शित करता है।

क्वॉइल तथा कॉन्टेक्ट लैडर लॉजिक इन्स्ट्रक्शन सेट के बेसिक सिम्बल होते हैं। कॉन्टेक्ट सिम्बल को दी गयी रंग कन्डीशन पर प्रोग्राम को रन किया जाता है, जो लो-आउटपुट को निर्धारित करता है तथा उसे कन्ट्रोल भी करता है। सभी निकलने वाले आउटपुट को क्वॉइल सिम्बल द्वारा प्रदर्शित किया जाता है। प्रत्येक कॉन्टेक्ट तथा क्वॉइल को एड्रेस नम्बर से सम्पादित किया जाता है। कॉन्टेक्ट को सीरीज तथा पैरेलल तरीके से स्पेसीफाइ किया जाता है जो एक निश्चित आउटपुट को कन्ट्रोल करता है।

## इन्स्ट्रक्शन का वर्गीकरण (Classification of Instructions)

इन्स्ट्रक्शन को निम्नलिखित प्रकार से वर्गीकृत किया जा सकता है—

**(i) बिट टाइप इन्स्ट्रक्शन्स** (Bit Type Instructions)—इस प्रकार के इन्स्ट्रक्शनों द्वारा बिट टाइप डाटा के ऑपरेशन को परफॉर्म किया जाता है।

**(ii) टाइमर इन्स्ट्रक्शन्स** (Timer Instructions)—इस प्रकार के इन्स्ट्रक्शन में टाइमर इन्स्ट्रक्शन के द्वा आवश्यकतानुसार टाइम डिले प्रदान किया जाता है।

**(iii) काउन्टर इन्स्ट्रक्शन्स** (Counter Instructions)—इस प्रकार के इन्स्ट्रक्शन का उपयोग वस्तुओं या उत्पाद की गिनती के लिए किया जाता है।

**(iv) लॉजिकल इन्स्ट्रक्शन्स** (Logical Instructions)—इस प्रकार के इन्स्ट्रक्शन में AND, OR, NOT त XOR आदि लॉजिकल इन्स्ट्रक्शनों का उपयोग किया जाता है जो इस प्रकार के ऑपरेशन को एक निश्चित फॉर्मेट प्रदर्शित करता है।

**(v) कम्पेरीजन इन्स्ट्रक्शन्स** (Comparison Instructions)—इस प्रकार के इन्स्ट्रक्शन में डाटा ऑपरेन्ड क कम्पेरीजन इन्स्ट्रक्शन द्वारा कम्पेयर किया जाता है।

**(vi) मैथ इन्स्ट्रक्शन्स** (Math Instructions)—इस प्रकार के इन्स्ट्रक्शन द्वारा अर्थमैटिक ऑपरेशनों को इनपु डाटा या ऑपरेन्ड पर परफॉर्म किया जाता है।

**(vii) सिक्वेन्सर इन्स्ट्रक्शन्स** (Sequencer Instructions)—इस प्रकार के इन्स्ट्रक्शन में आउटपुट डिवाइस किसी डाटा को क्रमबद्ध तरीके से लागू किया जाता है।

**(viii) डाटा ट्रांसफर इन्स्ट्रक्शन्स** (Data Transfer Instructions)—इस प्रकार के इन्स्ट्रक्शन में डाटा को एक मैमोरी लोकेशन से दूसरे मैमोरी लोकेशन में ट्रांसफर/कॉपी किया जाता है।

**(ix) PID कन्ट्रोल इन्स्ट्रक्शन्स** (PID Control Instructions)—यह इन्स्ट्रक्शन एक PID नियन्त्रक के रूप कार्य करता है तथा नियन्त्रक सिग्नल उत्पन्न करता है।

**(x) एडवांस मैथ इन्स्ट्रक्शन्स** (Advance Math Instructions)—इन निर्देशों की सहायता से डाटा की स्केल वर्गमूल ऑपरेशन, एनकोडिंग तथा डिकोडिंग आदि ऑपरेशन परफॉर्म किये जाते हैं।

**(xi) बिट शिफ्ट इन्स्ट्रक्शन्स** (Bit Shift Instructions)—इस प्रकार के इन्स्ट्रक्शनों द्वारा बिट बाइस डाटा क दायें से बायें शिफ्ट किया जाता है।

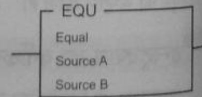
**(xii) ब्रांचिग इन्स्ट्रक्शन्स** (Branching Instructions)—इस प्रकार के इन्स्ट्रक्शन द्वारा प्रोग्राम कन्ट्रोल क विभिन्न रंग/लोकेशन में ट्रांसफर किया जाता है।

**(xiii) इनपुट/आउटपुट इन्स्ट्रक्शन्स** (Input/Output Instructions)—इस प्रकार के इन्स्ट्रक्शन में किस डिवाइस को इनपुट डाटा दिया जाता है तथा उस डाटा को आउटपुट डिवाइस से प्राप्त किया जाता है। इस प्रकार इनपुट/आउटपुट इन्स्ट्रक्शन को किसी महत्वपूर्ण फंक्शन के लिए उपयोग किया जाता है।

## कम्पेरीजन इन्स्ट्रक्शन (Comparison Instructions)

PLC में डाटा को कम्पेरीजन करने की क्षमता होती है। इस प्रकार से PLC इनपुट डिवाइस से प्राप्त डाटा त फाइल से प्राप्त डाटा को कम्पेयर कर सकती है। इसके आउटपुट इन्स्ट्रक्शन सत्य या असत्य होते हैं। यदि कम्पेरीज कन्डीशन संतुष्ट होती है तब आउटपुट सत्य होता है अन्यथा असत्य होता है। इस प्रकार से सामान्य कम्पेरीजन में टेस की गई वेल्यू की सहायता से रंग को इनर्जाइज्ड या डीइनर्जाइज्ड किया जाता है। PLC में उपयोग किये जाने वाले प्रमुख कम्पेरीजन इन्स्ट्रक्शन्स निम्नलिखित हैं—

**(i) इक्वल** (Equal)—इसे EQU निमोनिक्स से प्रदर्शित किया जाता है। इस इन्स्ट्रक्शन में दो वेल्यू की इक्वलिटी को ज्ञात किया जाता है। यदि दो वेल्यू सोर्स A तथा सोर्स B में कम्पेयर किया जाता है। यदि दोनों बराबर हैं तब उन्हें किसी अन्य मैमोरी लोकेशन में स्टोर किया जाता है तब सोर्स A का एड्रेस हमेशा वर्ड में होगा लेकिन सोर्स B का एड्रेस वर्ड या कॉन्स्टेन्ट वेल्यू में होगा। यदि सोर्स A तथा सोर्स B दोनों बराबर हैं तब इन्हें एक निश्चित फॉर्मेट में स्टोर किया जाता है तब रंग आउटपुट सत्य होगा अन्यथा इसका आउटपुट असत्य होगा। चित्र 3.1 में इक्वल

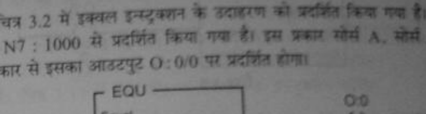



चित्र 3.1 : इक्वल इन्स्ट्रक्शन का फॉर्मेट



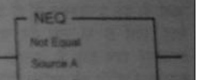
इन्स्ट्रक्शन के फॉर्मेट को प्रदर्शित किया गया है। इस चित्र में प्रथम लाइन में EQU के द्वारा फंक्शन के इन्स्ट्रक्शन को प्रदर्शित किया जाता है। इसमें सोर्स A तथा सोर्स B को मैमोरी एड्रेस के लिए स्पेसीफाइ किया जाता है, जहाँ पर डाटा स्टोर होता है।

उदाहरण के लिए, चित्र 3.2 में इक्वल इन्स्ट्रक्शन के उदाहरण को प्रदर्शित किया गया है। इसमें सोर्स A को N7 : 1000 तथा सोर्स B को N7 : 1000 से प्रदर्शित किया गया है। इस प्रकार सोर्स A, सोर्स B के बराबर हुआ अतः कन्डीशन संतुष्ट है। इस प्रकार से इसका आउटपुट O : 0/0 पर प्रदर्शित होगा।




चित्र 3.2

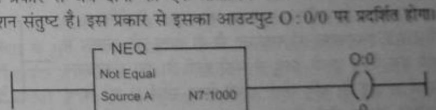
**(ii) नॉट इक्वल** (Not Equal)—नॉट इक्वल इन्स्ट्रक्शन को NEQ निमोनिक से प्रदर्शित किया जाता है। इसमें NEQ इन्स्ट्रक्शन की सहायता से दो वेल्यू को कम्पेयर किया जाता है जो बराबर नहीं है। इन दो वेल्यू को कंपेयर करके अलग मैमोरी लोकेशन पर स्टोर किया जाता है जिन्हें सोर्स A तथा सोर्स B के नाम से जाना जाता है। इस प्रकार से सोर्स A का एड्रेस, वर्ड के फार्मेट में होता है तथा सोर्स B का एड्रेस, वर्ड या कॉन्स्टेन्ट के फॉर्मेट में होता है। इस प्रकार से यदि सोर्स A तथा सोर्स B में स्टोर वेल्यू बराबर नहीं है तब इसका रंग आउटपुट सत्य होता है यदि सोर्स वेल्यू बराबर है तब यह असत्य होता है। चित्र 3.3 में नॉट इक्वल के फार्मेट को दर्शाया गया है।




चित्र 3.3 : नॉट इक्वल इन्स्ट्रक्शन का फॉर्मेट

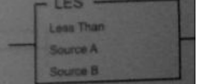
इस चित्र में प्रथम लाइन में not equal इन्स्ट्रक्शन के फंक्शन को प्रदर्शित किया गया है। इसमें सोर्स A तथा सोर्स B को मैमोरी एड्रेस के लिए स्पेसीफाइ किया गया है, जहाँ पर डाटा स्टोर होता है।

उदाहरण के लिए, चित्र 3.4 में नोट इक्वल इन्स्ट्रक्शन के उदाहरण को प्रदर्शित किया गया है। इसमें नॉट इक्वल इन्स्ट्रक्शन के द्वारा सोर्स A को N7 : 1000 के द्वारा प्रदर्शित किया गया है तथा सोर्स B को N7 : 900 के द्वारा प्रदर्शित किया गया है। इस प्रकार से जब दोनों को एक प्रोग्रामिंग फॉर्मेट में लाया जाता है तब यह बराबर नहीं होते हैं। इस प्रकार से इसकी कन्डीशन संतुष्ट है। इस प्रकार से इसका आउटपुट O : 0/0 पर प्रदर्शित होगा।




चित्र 3.4

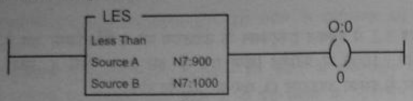
**(iii) लेस दैन** (Less Than)—लेस दैन इन्स्ट्रक्शन को LES निमोनिक्स से दर्शाया जाता है। इसमें LES इन्स्ट्रक्शन की सहायता से दो वेल्यू को आपस में कम्पेयर किया जाता है जिसमे एक वेल्यू दूसरी वेल्यू से छोटी होती है। इस प्रकार के इन्स्ट्रक्शन में सोर्स A तथा सोर्स B का उपयाग इनपुट के रूप में किया जाता है। यदि सोर्स A में स्टोर वेल्यू सोर्स B से कम है तो रंग आउटपुट सत्य होगा तथा यह वेल्यू कम नहीं है तो रंग आउटपुट असत्य होगा। चित्र 3.5 में लेस दैन इन्स्ट्रक्शन के फॉर्मेट को दर्शाया गया है




चित्र 3.5 : लेस दैन इन्स्ट्रक्शन का फॉर्मेट

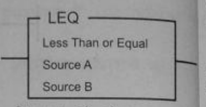
उदाहरण के लिए, चित्र 3.6 मे लेस दैन इन्स्ट्रक्शन के उदाहरण को प्रदर्शित किया गया है। इसमें लेस दैन इन्स्ट्रक्शन के द्वारा सोर्स A को N7 : 900 के द्वारा प्रदर्शित किया गया है तथा सोर्स B को N7 : 1000 से प्रदर्शित

किया गया है। इस प्रकार से जब दोनों सोर्सों को एक प्रोग्राम फॉर्मेट में लाया जाता है तब एक सोर्स दूसरे सोर्स से छोटा होता है। इस प्रकार से इसकी कन्डीशन संतुष्ट होती है तथा इसके आउटपुट को O : 0/0 पर प्रदर्शित किया जाता है।



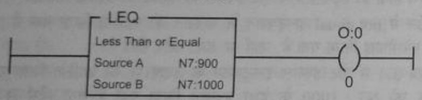

चित्र 3.6

**(iv) लेस दैन या इक्वल** (Less Than or Equal)—लेस दैन या इक्वल इन्स्ट्रक्शन को LEQ निमोनिक्स से दर्शाया जाता है। इसमें LEQ इन्स्ट्रक्शन की सहायता से दो वेल्यू को आपस कम्पेयर किया जाता है जिसमें एक वेल्यू दूसरी वेल्यू से कम होती है जिसे लॉजिकल प्रोग्रामिंग फॉर्मेट की सहायता से इक्वल में बदला जाता है। इस इन्स्ट्रक्शन के दो इनपुट सोर्स A तथा सोर्स B होते हैं। यदि सोर्स A की वेल्यू सोर्स B से लेस दैन या इक्वल है तब कन्डीशन संतुष्ट होती है तथा रंग आउटपुट सत्य होता है अन्यथा असत्य होता है। चित्र 3.7 में लेस दैन या इक्वल के इन्स्ट्रक्शन फॉर्मेट को प्रदर्शित किया गया है।



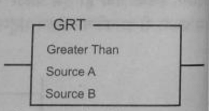

चित्र 3.7 : लेस देन या इक्वल इन्स्ट्रक्शन का फॉर्मेट

उदाहरण के लिए, चित्र 3.8 में लेस दैन या इक्वल के उदाहरण को दर्शाया गया है। इसमें लेस दैन या इक्वल इन्स्ट्रक्शन के द्वारा सोर्स A को N7 : 900 से तथा सोर्स B को N7 : 1000 से प्रदर्शित किया गया है। इस प्रकार से जब दोनों सोर्सों में एक प्रोग्रामिंग फॉर्मेट का उपयोग किया जाता है तब इसमें एक वेल्यू दूसरी वेल्यू से छोटी या बराबर होती है। इसके आउटपुट को O : 0/0 पर प्रदर्शित किया जाता है।

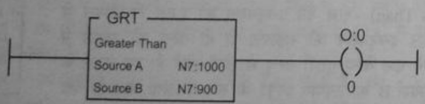



चित्र 3.8

**(v) ग्रेटर दैन** (Greater Than)—ग्रेटर दैन इन्स्ट्रक्शन को GRT निमोनिक्स से प्रदर्शित किया जाता है। इसमें GRT इन्स्ट्रक्शन की सहायता से दो वेल्यू को आपस में कम्पयेर किया जाता है, जिसमें एक वेल्यू दूसरी वेल्यू से बड़ी होती है। इस इन्स्ट्रक्शन में दो इनपुट सोर्स A तथा सोर्स B होते हैं। यदि सोर्स A की वेल्यू सोर्स B की वेल्यू से बड़ी है तब इसकी कन्डीशन संतुष्ट होती है तथा रंग आउटपुट सत्य होता है अन्यथा असत्य होता है। चित्र 3.9 में ग्रेटर दैन इन्स्ट्रक्शन के फॉर्मेट को प्रदर्शित किया गया है।




चित्र 3.9 : ग्रेटर दैन इन्स्ट्रक्शन का फॉर्मेट

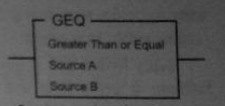



चित्र 3.10

उदाहरण के लिए, चित्र 3.10 में ग्रेटर दैन के उदाहरण को प्रदर्शित किया गया है। इसमें GRT इन्स्ट्रक्शन के द्वारा सोर्स A को N7 : 1000 से तथा सोर्स B को N7 : 900 से प्रदर्शित किया गया है। इस प्रकार से जब दोनों सोर्सों में एक निश्चित प्रोग्रामिंग फॉर्मेट का उपयोग किया जाता है तब इसमें एक वेल्यू दूसरी वेल्यू से बड़ी होती है तथा इसके आउटपुट को O : 0/0 पर प्रदर्शित किया जाता है।

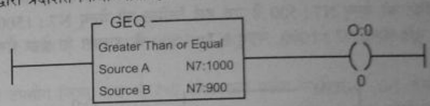


**(vi) ग्रेटर दैन या इक्वल** (Greater Than or Equal)—ग्रेटर दैन या इक्वल इन्स्ट्रक्शन को GEQ निमोनिक्स की सहायता से प्रदर्शित किया जाता है। इसमें GEQ इन्स्ट्रक्शन की सहायता से दो को एक दूसरे से कम्पयेर किया जाता है जिसमें एक वेल्यू दूसरी वेल्यू से बड़ी या बराबर होती है। इसमें एक वेल्यू को दूसरी वेल्यू के बराबर प्रदर्शित करने के लिए लॉजिकल इन्स्ट्रक्शन का उपयोग किया जाता है। इसमें यदि सोर्स A सोर्स B से ग्रेटर देन या इक्वल है तब इन्स्ट्रक्शन लॉजिकली सत्य होता है अन्यथा असत्य होता है। चित्र 3.11 में ग्रेटर दैन या इक्वल के इन्स्ट्रक्शन फॉर्मेट को दर्शाया गया है।



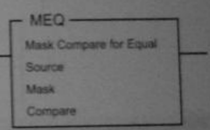

चित्र 3.11 : ग्रेटर दैन या इक्वल इन्स्ट्रक्शन का फॉर्मेट

उदाहरण के लिए, चित्र 3.12 में ग्रेटर दैन या इक्वल के उदाहरण को प्रदर्शित किया गया है। इसमें ग्रेटर दैन या इक्वल इन्स्ट्रक्शन के द्वारा सोर्स A को N7 : 1000 तथा सोर्स B को N7 : 900 से प्रदर्शित किया गया है। इसमें एक निश्चित प्रोग्रामिंग फॉर्मेट की सहातया से एक वेल्यू को दूसरी वेल्यू से बड़ी या बराबर प्रदर्शित की जाती है। इसके आउटपुट को O : 0/0 के द्वारा प्रदर्शित किया जाता है।

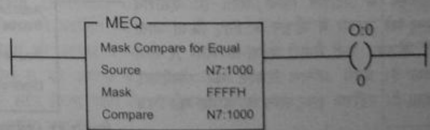



चित्र 3.12

**(vii) इक्वल के लिए मास्कड कम्पेरीजन** (Masked Comparison for Equal)—मास्कड कम्पेरीजन को इक्वल की सहायता से दर्शाने के लिए MEQ निमोनिक्स का उपयोग किया जाता है। यह इन्स्ट्रक्शन उन दो वेल्यू के पोर्शन को टेस्ट करता है जो आपस में बराबर हैं। इसके द्वारा सोर्स एड्रेस के 16-बिट डाटा को रिफरेन्स एड्रेस के 16-बिट डाटा से मास्क द्वारा कम्पेयर किया जाता है। इस इन्स्ट्रक्शन के द्वारा डाटा के उस पोर्शन को सेपरेट किया जाता है जिसे वर्ड द्वारा मास्क करना है। सोर्स उस एड्रेस की वेल्यू होती है जिसे किसी अन्य वेल्यू से कम्पयेर किया जाता है। अतः इस प्रकार से मास्क वह एड्रेस है जिसकी सहायता से मास्क के एड्रेस को डाटा में मूव कराया जाता है। इसमें मास्क हेक्साडेसीमल वेल्यू भी हो सकती है। इस प्रकार इसको रिफरेन्स वेल्यू की सहायता से इन्टीजर वेल्यू से कम्पेयर किया जाता है। इस प्रकार यदि 16-बिट का डाटा सोर्स एड्रेस के कम्पेयर एड्रेस के बराबर है तब इस इन्स्ट्रक्शन को सत्य माना जाता है। इस इन्स्ट्रक्शन की कन्डीशन जब सत्य होती है तब यह एड्रेस को मिसमैच करता है। चित्र 3.13 में इक्वल के लिए मास्क कम्पेरीजन के फॉर्मेट को दर्शाया गया है।




चित्र 3.13 : मास्क कम्पेरीजन इक्वल इन्स्ट्रक्शन का फॉर्मेट

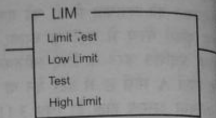



चित्र 3.14

उदाहरण के लिए, चित्र 3.14 में इक्वल के लिए मास्क कम्पेयर के उदाहरण को दर्शाया गया है। इसमें यदि सोर्स N7 : 1000 को 16-बिट के लिए मास्क किया जाता है तथा इसकी मास्क वेल्यू FFFFH हेक्साडेसीमल मे है तब इसे कम्पेयर एड्रेस N7 : 1000 पर प्रदर्शित किया जाता है जो 16-बिट एड्रेस का होता है। इस इन्स्ट्रक्शन में एक प्रोग्रामिंग

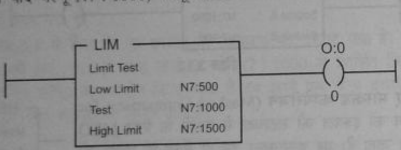
फॉर्मेंट का उपयोग किया जाता है जिसमें एक वेल्यू की दूसरी वेल्यू से मास्क किया जाता है तथा इसके आउटपुट को O :0/0 पर प्रदर्शित किया जाता है।

**(viii) लिमिट टेस्ट** (Limit Test)—लिमिट टेस्ट इन्स्ट्रक्शन को LIM निमोनिक्स से प्रदर्शित किया जाता है इसमें LIM इन्स्ट्रक्शन की सहायता से दो वेल्यू की लिमिट को चैक किया जाता है। इसमें एक वेल्यू की लिमिट रेन्ज दूसरी वेल्यू पर निर्भर करती है। इसमे लो लिमिट तथा हाई लिमिट वेल्यू वर्ड एड्रेस या कॉन्स्टेन्ट पर होती हैं। इसमें LIM इन्स्ट्रक्शन को एक स्पेसीफाइड रेन्ज में कम्पेयर किया जाता है। इस प्रकार से यदि टेस्ट की वेल्यू लो लिमिट से बड़ी या बराबर होती है तब यह हाई वेल्यू को लो वेल्यू के बराबर प्रदर्शित करता है। इस प्रकार रंग आउटपुट सत्य होता है अन्यथा असत्य होता है। चित्र 3.15 में लिमिट टेस्ट के इन्स्ट्रक्शन फॉर्मेंट को प्रदर्शित किया गया है।



**चित्र 3.15 : लिमिट टेस्ट इन्स्ट्रक्शन का फॉर्मेट**

उदाहरण के लिए, चित्र 3.16 में लिमिट टेस्ट के उदाहरण को दर्शाया गया है। इस प्रकार से यदि टेस्ट की वेल्यू N7 : 1000 है तथा लो लिमिट की वेल्यू N7 : 500 है एवं हाई लिमिट की वेल्यू N7 : 1500 है तब इसका आउटपुट O : 0/0 पर प्रदर्शित होगा। यदि वेल्यू N7 : 1000, वेल्यू N7 : 1500 के इक्वल या ग्रेटर दैन है तब इसकी कन्डीशन संतुष्ट होती है।

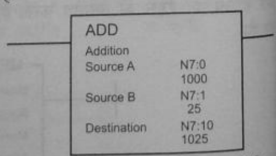


**चित्र 3.16**

## मैथमैटिकल इन्स्ट्रक्शन (Mathematical Instruction)

PLC में मैथमैटिकल इन्स्ट्रक्शनों को डाटा कम्प्यूटेशन इन्स्ट्रक्शन भी कहते हैं। अर्थमैटिक या मैथ फंक्शन की कॉन्स्टेन्ट वेल्यू को वर्ड या रजिस्टर में स्टोर किया जाता है तथा उससे सम्बन्धित ऑपरेशनों को परफॉर्म किया जाता है। इस प्रकार के इन्स्ट्रक्शनों द्वारा विभिन्न प्रकार के ऑपरेशन, जैसे—एडीशन, सबट्रेक्शन, मल्टीप्लिकेशन तथा डिवीजन आदि परफॉर्म किये जाते हैं। इन इन्स्ट्रक्शनों का एक्जीक्यूशन टाइम प्रोसेसर की प्रोसेसिंग क्षमता पर निर्भर करता है। मैथमैटिकल इन्स्ट्रक्शन के अन्तर्गत आने वाले प्रमुख इन्स्ट्रक्शन्स निम्नलिखित हैं—

**(i) एडीशन इन्स्ट्रक्शन** (Addition Instruction)—एडीशन इन्स्ट्रक्शन को ADD निमोनिक्स से प्रदर्शित किया जाता है। एडीशन इन्स्ट्रक्शन को उपयोग दो वेल्यू को आपस में जोड़ने के लिए किया जाता है। इस प्रकार से दो वेल्यू को जोड़कर उन्हें मैमोरी एड्रेस के सोर्स A तथा सोर्स B में स्टोर किया जाता है। इसमें अन्तिम रिजल्ट को स्पेसीफाइ डेस्टीनेशन पर स्टोर किया जाता है। एडीशन इन्स्ट्रक्शन के फॉर्मेट को चित्र 3.17 में दर्शाया गया है।



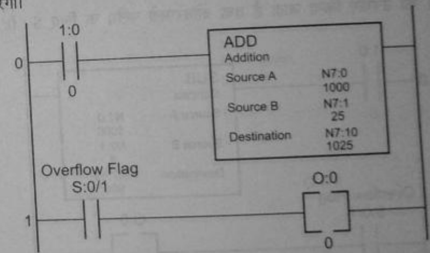
**चित्र 3.17 : एडीशन इन्स्ट्रक्शन का फॉर्मेट**

एडीशन इन्स्ट्रक्शन में अर्थमैटिक स्टेटस बिट का उपयोग किया जाता है जो एडीशन से सम्बन्धित ऑपरेशनों को एक निश्चित फॉर्मेट में प्रदर्शित करता है। एडीशन इन्स्ट्रक्शन के उपयोग में लाये जाने वाली प्रमुख अर्थमैटिक स्टेटस बिट तथा उनके फंक्शन नीचे सारणी में दिये गये हैं—



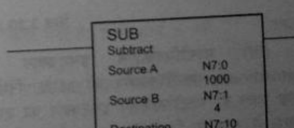
| क्र० सं० | अर्थमैटिक स्टेटस बिट | स्टेटस | फंक्शन |
|---|---|---|---|
| (i) | कैरी (C) | S : 0/0 | यदि कैरी जनरेट होती है तो 1 सेट करना अन्यथा 0 सेट करना। |
| (ii) | ओवरफ्लो (OV) | S : 0/1 | जब रिजल्ट फिट नहीं है या डेस्टीनेशन एड्रेस ओवरफ्लो है तब 1 सेट करना। |
| (iii) | जीरो (Z) | S : 0/2 | यदि मैथ रिजल्ट 0 के बराबर है तब 1 सेट करना। |
| (iv) | साइन (S) | S : 0/3 | जब मैथ इन्स्ट्रक्शन रिजल्ट निगेटिव (या शून्य से कम) है तब 1 सेट करना। |

**उदाहरण**—चित्र 3.18 में एडीशन इन्स्ट्रक्शन के उदाहरण को दर्शाया गया है। इसमें सोर्स A को $\frac{N7:0}{1000}$ से तथा सोर्स B को $\frac{N7:0}{25}$ से प्रदर्शित किया गया है। जब दोनों के बीच एडीशन ऑपरेशन परफॉर्म करते हैं तब उसके रिजल्ट को $\frac{N7:10}{1025}$ पर प्रदर्शित किया जाता है तथा इसके आउटपुट को O : 0/0 पर प्रदर्शित किया जाता है। इसमें इनपुट के लिए डेस्टीनेशन I : 0/0 का उपयोग किया जाता है तथा ओवरफ्लो फ्लैग के लिए S : 0/1 डेस्टीनेशन का उपयोग किया जाता है। इस प्रकार से यदि रंग कन्डीशन सत्य है तब ADD इन्स्ट्रक्शन सोर्स A तथा सोर्स B को जोड़ेगा तथा रिजल्ट को डेस्टीनेशन पर स्टोर करेगा।



**चित्र 3.18 : एडीशन इन्स्ट्रक्शन का उदाहरण**

**(ii) सबट्रेक्ट इन्स्ट्रक्शन** (Subtract Instruction)—सबट्रेक्ट इन्स्ट्रक्शन को SUB निमोनिक्स से प्रदर्शित किया जाता है। सबट्रेक्शन इन्स्ट्रक्शन का उपयोग दो वेल्यू को आपस में घटाने में किया जाता है। इस प्रकार सोर्स A तथा सोर्स B को आपस में घटाकर उसके रिजल्ट को डिफरेन्ट मैमोरी एड्रेस में स्टोर किया जाता है। सबट्रेक्शन इन्स्ट्रक्शन के फॉर्मेट को चित्र 3.19 में दर्शाया गया है।



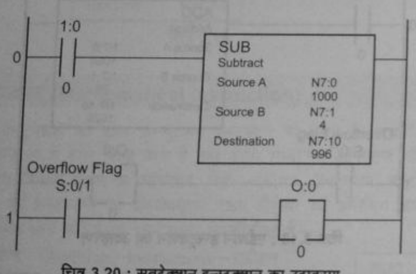
**चित्र 3.19 : सबट्रेक्शन इन्स्ट्रक्शन का उदाहरण**

इस प्रकार से यदि इसकी रंग कन्डीशन सत्य है तब SUB इन्स्ट्रक्शन सोर्स A में से सोर्स B को घटाता है तथा रिजल्ट को डेस्टीनेशन पर स्टोर करता है।

सबट्रेक्शन इन्स्ट्रक्शन में अर्थमैटिक स्टेटस बिट का उपयोग किया जाता है जो सबट्रेक्शन से सम्बन्धित विभिन्न प्रकार के ऑपरेशनों को परफॉर्म करता है। सबट्रेक्शन इन्स्ट्रक्शन में उपयोग किये जाने वाली प्रमुख अर्थमैटिक स्टेटस बिट तथा उनके फंक्शन नीचे सारणी में दर्शाये गये हैं—

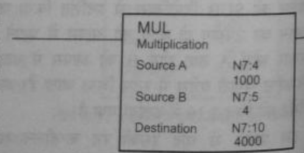
| क्र० सं० | अर्थमैटिक स्टेटस बिट | स्टेटस | फंक्शन |
|---|---|---|---|
| (i) | कैरी (C) | S : 0/1 | यदि कैरी जनरेट होती है तब सेट करना अन्यथा रीसेट करना। |
| (ii) | ओवरफ्लो (OV) | S : 0/1 | यदि ओवरफ्लो डेस्टीनेशन को डिटेक्ट करता है तब उसे सेट करना अन्यथा रीसेट करना। |
| (iii) | जीरो (Z) | S : 0/2 | यदि रिजल्ट जीरो है तब सेट करना अन्यथा रीसेट करना। |
| (iv) | साइन (S) | S : 0/3 | यदि रिजल्ट निगेटिव है तब सेट करना अन्यथा रीसेट करना। |

**उदाहरण**—चित्र 3.20 में सबट्रेक्शन के उदाहरण को प्रदर्शित किया गया है। इसमें सोर्स A को $\underset{1000}{N7:0}$ से त सोर्स B को $\underset{4}{N7:1}$ से प्रदीत किया गया है। जब सोर्स A में से सोर्स B को घटाया जाता है तब इसका रिज डेस्टीनेशन $\underset{996}{N7:10}$ पर प्राप्त होता है। इस प्रकार से इसके आउटपुट को O : 0/1 पर प्रदर्शित किया गया है। इसमें इन डेस्टीनेशन के लिए I : 0/1 का उपयोग किया जाता है तथा ओवरफ्लो फ्लैग के लिए S : 0/1 डेस्टीनेशन का उपयो किया जाता है।




चित्र 3.20 : सबट्रेक्शन इन्स्ट्रक्शन का उदाहरण

**(iii) मल्टीप्लिकेशन इन्स्ट्रक्शन** (Multiplication Instruction)—मल्टीप्लिकेशन को MUL निमोनिक्स से प्रदर्शित किया जाता है। मल्टीप्लिकेशन इन्स्ट्रक्शन का उपयोग दो वेल्यू को आपस में गुणा के लिए किया जाता है। इस प्रकार सोर्स A तथा सोर्स B का आपस में गुणा करके उसके रिजल्ट को डिफरेन्ट मैमोरी एड्रेस में स्टोर किया जाता है। मल्टीप्लिकेशन इन्स्ट्रक्शन के फॉर्मेट को चित्र 3.21 में दर्शाया गया है। इस प्रकार से जब रंग कन्डीशन सत्य होती है तब सोर्स A तथा सोर्स B का आपस में गुणा करके उसके रिजल्ट को डेस्टीनेशन पर दर्शाया जाता है।



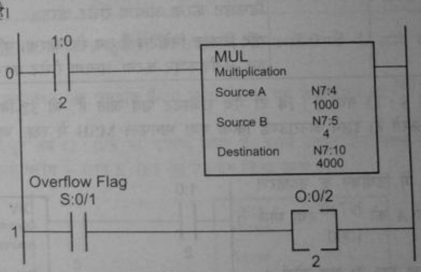

चित्र 3.21 : मल्टीप्लिकेशन इन्स्ट्रक्शन का फॉर्मेट



मल्टीप्लिकेशन इन्स्ट्रक्शन के अर्थमैटिक स्टेटस बिट का उपयोग किया जाता है जो मल्टीप्लिकेशन से सम्बन्धित विभिन्न प्रकार के फंक्शनों को परफॉर्म करता है। मल्टीप्लिकेशन इन्स्ट्रक्शन में उपयोग किये जाने वाले अर्थमैटिक स्टेटस बिट तथा उसके फंक्शन नीचे सारणी में दर्शाये गये हैं—

| क्र० सं० | अर्थमैटिक स्टेटस बिट | स्टेटस | फंक्शन |
|---|---|---|---|
| (i) | कैरी (C) | S : 0/0 | इसमें कैरी स्टेटस बिट हमेशा रीसेट रहती है। |
| (ii) | ओवरफ्लो (OV) | S : 0/1 | यदि ओवरफ्लो डेस्टीनेशन को डिटेक्ट करता है तब यह सेट होती है अन्यथा रीसेट होती है। |
| (iii) | जीरो (Z) | S : 0/2 | यदि रिजल्ट जीरो है तब सेट करना (यदि ओवरफ्लो फ्लैग सेट है तब अनडिफाइन करना) अन्यथा रीसेट करना। |
| (iv) | साइन (S) | S : 0/3 | यदि रिजल्ट निगेटिव है तब सेट करना (यदि ओवरफ्लो फ्लैग सेट है तब अनडिफाइन करना) अन्यथा रीसेट करना। |

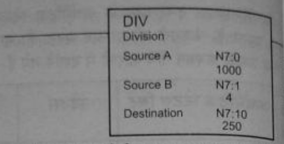
मल्टीप्लिकेशन इन्स्ट्रक्शन में S : 13 तथा S : 14 मैथ रजिस्टर पाये जाते हैं जो 32-बिट साइन इन्टीजर के द्वारा मल्टीप्लाइ ऑपरेशन को परफॉर्म करते हैं। इस प्रकार से S : 13 LSB तथा S : 14 MSB स्टेटस वर्ड को 32-बिट वेल्यू क इन्स्ट्रक्शन में रखा जाता है।




चित्र 3.22 : मल्टीप्लिकेशन इन्स्ट्रक्शन का उदाहरण

**उदाहरण**—चित्र 3.22 में मल्टीप्लिकेशन के उदाहरण को प्रदर्शित किया गया है। इसमें सोर्स A को $\underset{1000}{N7:4}$ तथा सोर्स B को $\underset{4}{N7:5}$ से प्रदर्शित किया गया है। जब सोर्स A को सोर्स B से मल्टीप्लाइ किया जाता है तब इसका रिजल्ट डेस्टीनेशन $\underset{4000}{N7:10}$ पर प्राप्त होता है। इस प्रकार से इसके आउटपुट को O : 0/2 से प्रदर्शित किया जाता है। इसमें इनपुट डेस्टीनेशन के लिए I : 0/2 का उपयोग किया जाता है तथा ओवरफ्लो फ्लैग के लिए S : 0/1 डेस्टीनेशन का उपयोग किया जाता है।

**(iv) डिवीजन इन्स्ट्रक्शन** (Division Instruction)—डिवीजन इन्स्ट्रक्शन को DIV निमोनिक्स से प्रदर्शित किया जाता है। डिवीजन इन्स्ट्रक्शन का उपयोग दो वेल्यू को आपस में डिवाइस करने के लिए किया जाता है। इस प्रकार से सोर्स A को सोर्स B से डिवाइड करते हैं तथा रिजल्ट को मैमोरी एड्रेस में स्टोर करते हैं। इस प्रकार से जब रंग कन्डीशन सत्य होती है तब सोर्स A को सोर्स B से डिवाइड किया जाता है तथा उसके रिजल्ट को डेस्टीनेशन पर दर्शाया जाता है। चित्र 3.23 में डिवीजन इन्स्ट्रक्शन के फॉर्मेट को दर्शाया जाता है।



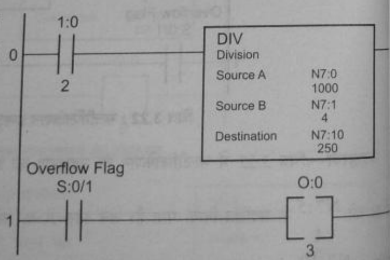

**चित्र 3.23 : डिवीजन इन्स्ट्रक्शन का फॉर्मेट**

डिवीजन इन्स्ट्रक्शन में अर्थमैटिक स्टेटस बिट का उपयोग किया जाता है जो डिवीजन से सम्बन्धित विधि फंक्शनों को परफॉर्म करता है। डिवीजन इन्स्ट्रक्शन में उपयोग किये जाने वाले प्रमुख अर्थमैटिक स्टेट तथा उनके फंक नीचे सारणी में दर्शाये गये हैं—

| क्र० सं० | अर्थमैटिक स्टेटस बिट | स्टेटस | फंक्शन |
|---|---|---|---|
| (i) | कैरी (C) | S : 0/0 | इसमें कैरी फ्लैग बिट हमेशा रीसेट रहती है। |
| (ii) | ओवरफ्लो (OV) | S : 0/1 | यदि ओवरफ्लो डेस्टीनेशन को डिटेक्ट करता है या जीरो से डिवाइ करता है, तब यह सेट होता है अन्यथा रीसेट होता है। |
| (iii) | जीरो (Z) | S : 0/2 | यदि रिजल्ट जीरो है तब सेट करना। यदि ओवरफ्लो फ्लैग सेट है त डिफाइन करना अन्यथा रीसेट करना। |
| (iv) | साइन (S) | S : 0/3 | यदि रिजल्ट निगेटिव है तब सेट करना। यदि ओवरफ्लो फ्लैग सेट तब अनडिफाइन करना अन्यथा रीसेट करना। |

डिवीजन इन्स्ट्रक्शन में S : 13 तथा S : 14 दो मैथ रजिस्टर पाये जाते हैं, जो 32-बिट साइन इन्टीजर के द्व डिवीजन ऑपरेशन परफॉर्म करते हैं। इसमें अनराउण्ड किया गया भागफल MSB में रखा जाता है तथा शेषफल LS में रखा जाता है।

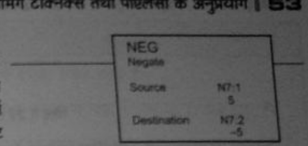
**उदाहरण**—चित्र 3.24 में डिवीजन के उदाहरण को दर्शाया गया है। इसमें सोर्स A को $\underset{1000}{N7:0}$ तथा सोर्स N को $\underset{4}{N7:1}$ से प्रदर्शित किया गया है। जब सोर्स A को सोर्स B से डिवाइड किया जाता है तब इसका रिजल्ट डेस्टीनेशन $\underset{250}{N7:10}$ पर प्राप्त होता है। इस प्रकार से इसके आउटपुट को O : 0/3 से प्रदर्शित किया जाता है। इसमें इनपुट डेस्टीनेशन के लिए I : 0/2 का उपयोग किया जाता है तथा आवेरफ्लो फ्लैग के लिए S : 0/1 डेस्टीनेशन का उपयोग किया जाता है।




**चित्र 3.24 : डिवीजन इन्स्ट्रक्शन का उदाहरण**



**(v) निगेट इन्स्ट्रक्शन** (Negate Instruction)—निगेट इन्स्ट्रक्शन को NEG निमोनिक्स द्वारा दर्शाया जाता है। निगेट इन्स्ट्रक्शन का उपयोग, सोर्स वेल्यू के साइन को बदलकर उसे डेस्टीनेशन पर प्राप्त किया जाता है। इस प्रकार से इसमें यदि सोर्स धनात्मक है तो वह ऋणात्मक हो जायेगा। चित्र 3.25 में निगेट इन्स्ट्रक्शन के फॉर्मेट को दर्शाया गया है।

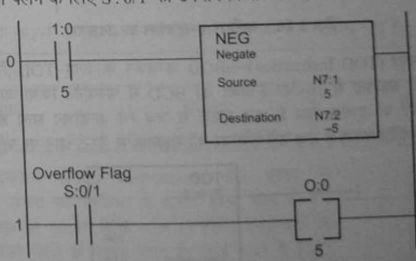



**चित्र 3.25 : निगेट इन्स्ट्रक्शन का फॉर्मेट**

निगेट इन्स्ट्रक्शन में अर्थमैटिक स्टेटस बिट का उपयोग किया जाता है जिसमें निगेट से सम्बन्धित विभिन्न प्रकार के ऑपरेशनों को परफॉर्म किया जाता है। निगेट इन्स्ट्रक्शन में उपयोग किये जाने वाले अर्थमैटिक स्टेटस बिट तथा उसके फंक्शन नीचे सारणी में दर्शाये गये हैं—

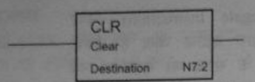
| क्र० सं० | अर्थमैटिक स्टेटस बिट | स्टेटस | फंक्शन |
|---|---|---|---|
| (i) | कैरी (C) | S : 0/0 | यदि कैरी उपस्थित है तो सेट करना अन्यथा रीसेट करना। |
| (ii) | ओवरफ्लो (OV) | S : 0/1 | यदि ओवरफ्लो फ्लैग उपस्थित है तो सेट करना अन्यथा रीसेट करना। |
| (iii) | जीरो (Z) | S : 0/2 | यदि रिजल्ट जीरो है तो सेट करना अन्यथा रीसेट करना। |
| (iv) | साइन (S) | S : 0/3 | यदि रिजल्ट निगेटिव है तब सेट करना अन्यथा रीसेट करना। |

**उदाहरण**—चित्र 3.26 में निगेट इन्स्ट्रक्शन के उदाहरण को दर्शाया गया है। इसमें सोर्स को $\underset{5}{N7:1}$ से दर्शाया गया है। जब इसमें निगेट इन्स्ट्रक्शन का उपयोग किया जाता है तब यह डेस्टीनेशन पर $\underset{-5}{N7:2}$ के रूप में प्राप्त होता है। इस प्रकार से इसके आउटपुट तब O : 0/5 पर दर्शाया जाता है। इसमें इनपुट डेस्टीनेशन के लिए I : 0/5 का उपयोग किया जाता है तथा ओवरफ्लो फ्लैग के लिए S : 0/1 का उपयोग किया जाता है।




**चित्र 3.26 : निगेट इन्स्ट्रक्शन का उदाहरण**

**(vi) क्लीयर इन्स्ट्रक्शन** (Clear Instructions)—क्लीयर इन्स्ट्रक्शन को CLR निमोनिक्स से प्रदर्शित किया जाता है। इसमें क्लीयर इन्स्ट्रक्शन का उपयोग किसी वर्ड को डेस्टीनेशन पर जीरो दर्शाने के लिए किया जाता है। इसमें एक निश्चित प्रोग्रामिंग फॉर्मेट का उपयोग करके इसके डेस्टीनेशन को सेट किया जाता है। चित्र 3.27 में क्लीयर इन्स्ट्रक्शन के फॉर्मेट को दर्शाया गया है।

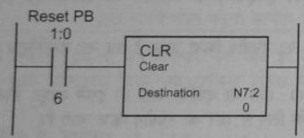



चित्र 3.27 : क्लीयर इन्स्ट्रक्शन का फॉर्मेट

क्लीयर इन्स्ट्रक्शन में अर्थमैटिक स्टेटस बिट का उपयोग किया जाता है जो क्लीयर इन्स्ट्रक्शन से सम्ब विभिन्न प्रकार के ऑपरेशनों को परफॉर्म करता है। क्लीयर इन्स्ट्रक्शन में उपयोग किये जाने वाले अर्थमैटिक स्टेटस तथा उनके फंक्शन नीचे सारणी में दर्शाये गये हैं—

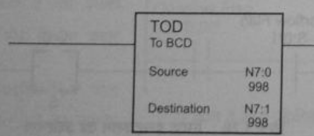
| क्र० सं० | अर्थमैटिक स्टेटस बिट | स्टेटस | फंक्शन |
|---|---|---|---|
| (i) | कैरी (C) | S : 0/0 | इसमें कैरी हमेशा रीसेट रहती है। |
| (ii) | ओवरफ्लो (OV) | S : 0/1 | इसमें ओवरफ्लो हमेशा रीसेट रहता है। |
| (iii) | जीरो (Z) | S : 0/2 | इसमें जीरो फ्लैग हमेशा रीसेट रहता है। |
| (iv) | साइन (S) | S : 0/3 | इसमें साइन फ्लैग हमेशा रीसेट रहता है। |

**उदाहरण**—चित्र 3.28 में क्लीयर इन्स्ट्रक्शन के उदाहरण को दर्शाया गया है। इसमें जब इनपुट I : 0/6 पर है तब रीसेट PB ऑन होती है तथा क्लीयर इन्स्ट्रक्शन कॉन्स्टेन्ट्स को $\frac{N7:2}{0}$ पर क्लीयर करता है।




चित्र 3.28 : क्लीयर इन्स्ट्रक्शन का उदाहरण

**(vii) TOD इन्स्ट्रक्शन** (TOD Instruction)—BCD कन्वर्जन के लिए TOD निमोनिक्स का उपयोग कि जाता है। इस इन्स्ट्रक्शन की सहायता से 16-बिट इन्टीजर को BCD में परिवर्तित किया जाता है। चित्र 3.29 में TO से BCD कन्वर्टर के फॉर्मेट को दर्शाया गया है। इस प्रकार से जब रंग कन्डीशन सत्य होती है तो TOD इन्स्ट्र सोर्स वेल्यू को BCD में परिवर्तित करता है तथा उसे रजिस्टर की सहातया से डेस्टीनेशन पर पहुँचाता है।




चित्र 3.29 : TOD इन्स्ट्रक्शन का फॉर्मेट

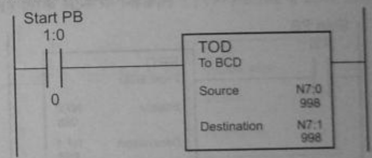
TOD से BCD कन्वर्टर में अर्थमैटिक स्टेटस बिट का उपयोग किया जाता है जो TOD इन्स्ट्रक्शन की सहा से विभिन्न प्रकार के ऑपरेशन को परफॉर्म करता है। TOD इन्स्ट्रक्शन में उपयोग किये जाने वाले स्टेटस बिट त उनके फंक्शन नीचे सारणी में दिये गये हैं—



| क्र० सं० | अर्थमैटिक स्टेटस बिट | स्टेटस | फंक्शन |
|---|---|---|---|
| (i) | कैरी (C) | S : 0/0 | इसमें कैरी हमेशा रीसेट रहती है। |
| (ii) | ओवरफ्लो (OV) | S : 0/1 | यदि रिजल्ट 9999 से बड़ा है तब BCD सेट होता है। |
| (iii) | जीरो (Z) | S : 0/2 | इसमें जब डेस्टीनेशन वेल्यू शून्य होती है तब यह सेट होता है। |
| (iv) | साइन (S) | S : 0/3 | इसमें यदि वर्ड सोर्स निगेटिव है तब यह सेट होता है अन्यथा रीसेट होता है। |

BCD कन्वर्टर के TOD इन्स्ट्रक्शन में मैथ रजिस्टर का उपयोग किया जाता है जो 5 डिजिट के BCD रिजल्ट को कन्वर्ट करता है। इसमें S : 13 तथा S : 14 मैथ रजिस्टर पाये जाते हैं। इसके स्टेटस वर्ड के S : 13 रजिस्टर में LSB वेल्यू तथा S : 14 रजिस्टर में MSB वेल्यू को 32-बिट में रखा जाता है।

**उदाहरण**—चित्र 3.30 में TOD से BCD कन्वर्टर के उदाहरण को दर्शाया गया है। इसमें सोर्स को $\frac{N7:0}{998}$ तथा डेस्टीनेशन को $\frac{N7:1}{998}$ से प्रदर्शित किया गया है। इस प्रकार से इनपुट I : 0/0 ऑन है तब TOD इन्स्ट्रक्शन 998 को बाइनरी फॉर्मेट में परिवर्तित करता है जिसे डेस्टीनेशन के N7 : 1 लोकेशन पर स्टोर किया जाता है।

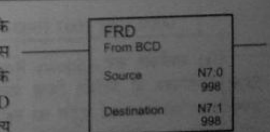



चित्र 3.30 : TOD इन्स्ट्रक्शन का उदाहरण

डेसीमल से बाइनरी तथा बाइनरी से BCD कन्वर्ट बिट को अग्र सारणी में दर्शाया गया है—

| N7 : 0 | N7 : 0 | N7 : 1 |
|---|---|---|
| Decimal | Binary | BCD |
| 998 | 0000 0011 1110 0110 | 0000 1001 1001 1000 |

**(viii) FRD इन्स्ट्रक्शन** (FRD Instruction)—इस प्रकार के इन्स्ट्रक्शन में किसी BCD संख्या को डेसीमल में कन्वर्ट किया जाता है। इस कन्वर्जन में FRD निमोनिक्स का उपयोग किया जाता है। FRD इन्स्ट्रक्शनों के द्वारा BCD वेल्यू को इन्टीजर वेल्यू में बदला जाता है। चित्र 3.31 में FRD इन्स्ट्रक्शन के फॉर्मेट को दर्शाया गया है। इस प्रकार जब रंग कन्डीशन सत्य होती है तब FRD इन्स्ट्रक्शन के द्वारा BCD वेल्यू को मैथ रजिस्टर या सोर्स इन्टीजर में बदला जाता है तथा उसे डेस्टीनेशन में स्टोर किया जाता है। यह TOD इन्स्ट्रक्शन के विपरीत होता है।

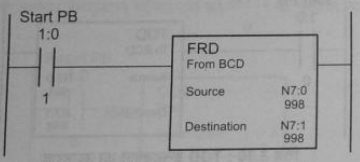



चित्र 3.31 : FRD इन्स्ट्रक्शन का फॉर्मेट

FRD इन्स्ट्रक्शन में अर्थमैटिक स्टेटस बिट का उपयोग किया जाता है जो FRD इन्स्ट्रक्शन की सहायता विभिन्न फंक्शनों को परफॉर्म करता है। FRD इन्स्ट्रक्शन में उपयोग किये जाने वाले अर्थमैटिक स्टेटस बिट तथा उनके फंक्शन नीचे सारणी में दिये गये हैं—

| क्र० सं० | अर्थमैटिक स्टेटस बिट | स्टेटस | फंक्शन |
|---|---|---|---|
| (i) | कैरी (C) | S : 0/0 | इसमें कैरी हमेशा रीसेट रहती है। |
| (ii) | ओवरफ्लो (OV) | S : 0/1 | इसमें नॉन-BCD वेल्यू सोर्स में है या कन्वर्ट की गयी वेल्यू 32767 से बड़ी है तब सेट होती है अन्यथा रीसेट होती है। |
| (iii) | जीरो (Z) | S : 0/2 | यदि डेस्टीनेशन वेल्यू शून्य होती है तब सेट होता है। |
| (iv) | साइन (S) | S : 0/3 | इसमें साइन हमेशा रीसेट रहता है। |

**उदाहरण**—चित्र 3.32 में FRD इन्स्ट्रक्शन के उदाहरण को दर्शाया गया है। इसमें सोर्स को N7:0 998 डेस्टीनेशन को N7:1 998 से प्रदश्रित किया गया है। इस प्रकार से जब इनपुट I : 0/1 ऑन होता है तब FRD इनस्ट्रक्शन 998 को बाइनरी फॉर्मेट में परिवर्तित करता है तथा उसे N7 : 1 लोकेशन पर स्टोर करता है।

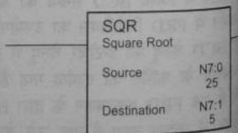



**चित्र 3.32 : FRD इन्स्ट्रक्शन का उदाहरण**

इस प्रकार BCD से बाइनरी और डेसीमल कन्वर्जन को अग्र सारणी में दर्शाया गया है।

| N7 : 0 | N7 : 1 | N7 : 1 |
|---|---|---|
| BCD | Binary | Decimal |
| 0000 1001 1001 1000 | 0000 0011 1110 0110 | 998 |

**(ix) स्क्वैअर रूट** (Square Root)—स्क्वैअर रूट इन्स्ट्रक्शन को SQR निमोनिक्स से प्रदर्शित किया जाता है। इसमें SQR इन्स्ट्रक्शन का उपयोग कर सोर्स के वर्ड एड्रेस का स्क्वैअर रूट ज्ञात किया जाता है तथा उसे डेस्टीनेशन पर दर्शाया जाता है। चित्र 3.33 में स्क्वैअर रुट के फॉर्मेट को दर्शाया गया है। इस प्रकार से जब रंग कन्डीशन सत्य होती है तब SQR इन्स्ट्रक्शन सोर्स के स्क्वैअर रूट को कैलकुलेट करता है तथा इन्टीजर रिजल्ट को डेस्टीनेशन में स्टोर करके रखता है।



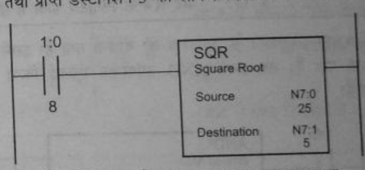

**चित्र 3.33 : स्क्वैअर रूट इन्स्ट्रक्शन का फॉर्मेट**



स्क्वैअर रूट इन्स्ट्रक्शन में अर्थमैटिक स्टेटस बिट का उपयोग किया जाता है जो SQR इन्स्ट्रक्शन से सम्बन्धित विभिन्न प्रकार के ऑपरेशनों को परफॉर्म करता है। SQR इन्स्ट्रक्शन में उपयोग किये जाने वाले अर्थमैटिक स्टेटस बिट तथा उनके फंक्शन नीचे सारणी में दिये गये हैं—

| क्र० सं० | अर्थमैटिक स्टेटस बिट | स्टेटस | फंक्शन |
|---|---|---|---|
| (i) | कैरी (C) | S : 0/0 | इसमें कैरी हमेशा रीसेट रहती है। |
| (ii) | ओवरफ्लो (OV) | S : 0/1 | इसमें ओवरफ्लो हमेशा रीसेट रहता है। |
| (iii) | जीरो (Z) | S : 0/2 | इसमें यदि डेस्टीनेशन वेल्यू शून्य रहती है तब यह सेट होता है। |
| (iv) | साइन (S) | S : 0/3 | इसमें साइन हमेशा रीसेट रहता है। |

**उदाहरण**—चित्र 3.34 में स्क्वैअर रूट के उदाहरण को दर्शाया गया है। इसमें सोर्स को N7:0 25 तथा डेस्टीनेशन को N7:1 5 से प्रदर्शित किया गया है। इस प्रकार से जब इसका इनपुट I : 0/8 ऑन होता है तब SQR इन्स्ट्रक्शन 25 के स्क्वैअर को कैलकुलेट करता है तथा प्राप्त डेस्टीनेशन 5 को लोकेशन N7 : 1 पर स्टोर करता है।

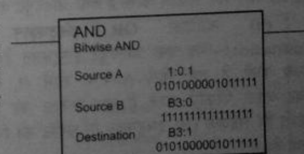



**चित्र 3.34 : स्क्वैअर रूट इन्स्ट्रक्शन का उदाहरण**

## लॉजिकल इन्स्ट्रक्शन (Logical Instruction)

लॉजिकल इन्स्ट्रक्शन के अन्तर्गत लॉजिकल AND, लॉजिकल OR, XOR, NOT तथा कम्पेयर आदि ऑपरेशन परफॉर्म किये जाते हैं। लॉजिकल इन्स्ट्रक्शन के डाटा को इनपुट डिवाइस से प्राप्त किया जाता है तथा उस डाटा को डाटा फाइल में लॉजिकली स्टोर किया जाता है तथा इसके रिजल्ट को डेस्टीनेशन में स्टोर किया जाता है। लॉजिकल इन्स्ट्रक्शन के अन्तर्गत आने वाले प्रमुख इन्स्ट्रक्शन्स निम्नलिखित हैं—

**(i) लॉजिक AND इन्स्ट्रक्शन** (Logic AND Instruction)—यह एक आउटपुट इन्स्ट्रक्शन है। जब रंग कन्डीशन सत्य होती है तब सोर्स A तथा सोर्स B, AND ऑपरेशन परफॉर्म करता है एवं रिजल्ट को डेस्टीनेशन पर स्टोर करता है। लॉजिक AND इन्स्ट्रक्शन के फॉर्मेट को चित्र 3.35 में दर्शाया गया है।




**चित्र 3.35 : लॉजिक AND इन्स्ट्रक्शन का फॉर्मेट**

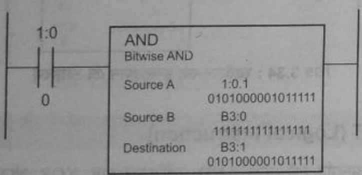
AND इन्स्ट्रक्शन के ऑपरेशन में सत्य सारणी का उपयोग किया जाता है। सत्य सारणी में सोर्स A तथा सोर्स B के ऑपरेशन को डेस्टीनेशन की सहायता से दर्शाया जाता है। AND इन्स्ट्रक्शन की सत्य सारणी नीचे दी गयी है—

| सोर्स *A* | सोर्स *B* | डेस्टीनेशन |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

AND इन्स्ट्रक्शन में मैथ फ्लैग का उपयोग विभिन्न ऑपरेशनों को परफॉर्म करने में किया जाता है। AN इन्स्ट्रक्शन के अन्तर्गत आने वाले मैथ फ्लैग तथा उसके फंक्शन नीचे सारणी में दिये गये हैं—

| क्र० सं० | मैथ फ्लैग | स्टेटस | फंक्शन |
|---|---|---|---|
| (i) | कैरी फ्लैग (C) | S : 0/0 | यदि कैरी उत्पन्न होती है तब कैरी फ्लैग सेट होता है अन्य रीसेट होता है। |
| (ii) | ओवरफ्लो फ्लैग (OV) | S : 0/1 | यदि ओवरफ्लो डेस्टीनेशन में उपलब्ध होता है, तब ओवरफ्ल फ्लैग सेट होता है अन्यथा रीसेट होता है। |
| (iii) | जीरो फ्लैग (Z) | S : 0/2 | यदि रिजल्ट जीरो है तब जीरो फ्लैग सेट होता है। |
| (iv) | साइन फ्लैग (S) | S : 0/3 | यदि रिजल्ट निगेटिव होता है तब साइन फ्लैग सेट होता है। |

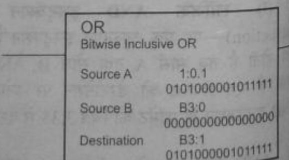
**उदाहरण**—चित्र 3.36 में AND इन्स्ट्रक्शन के उदाहरण को दर्शाया गया है। इसमें सोर्स A को I : 01 तथा स B को B3 : 0 से प्रदर्शित किया गया है। जब इसमें AND ऑपरेशन एप्लाई किया जाता है तब इसका आउट 'डेस्टीनेशन B3 : 1 पर प्राप्त होता है।




**चित्र 3.36 : लॉजिक AND इन्स्ट्रक्शन का उदाहरण**

इस प्रकार से जब इनपुट I : 0/0 ऑन होता है तब सोर्स A, I : 0.1S 16-बिट तथा सोर्स B, B3 : 0 16-बि AND ऑपरेशन परफॉर्म करता है तथा आउटपुट को डेस्टीनेशन पर स्टोर करता है।

**(ii) लॉजिक OR इन्स्ट्रक्शन** (Logic OR Instruction)—यह एक आउटपुट इन्स्ट्रक्शन है। जब रंग कन्डीशन सत्य होती है तब सोर्स A तथा सोर्स B, OR ऑपरेशन को बिट-वाइस परफॉर्म करते हैं तथा रिजल्ट को डेस्टीनेशन में स्टोर करते हैं। लॉजिक OR इन्स्ट्रक्शन के फॉर्मेट को चित्र 3.37 में दर्शाया गया है।




**चित्र 3.37 : लॉजिक OR इन्स्ट्रक्शन का फॉर्मेट**

OR इन्स्ट्रक्शन के ऑपरेशन में सत्य सारणी का उपयोग OR ऑपरेशन परफॉर्म करने के लिए किया जाता है। सत्य सारणी में सोर्स



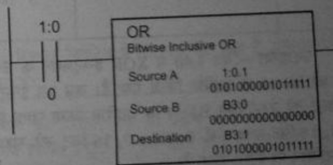
A तथा सोर्स B के ऑपरेशन को डेस्टीनेशन की सहायता से दर्शाया जाता है। OR इन्स्ट्रक्शन की सत्य सारणी नीचे दी गयी है—

| सोर्स *A* | सोर्स *B* | डेस्टीनेशन |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

OR इन्स्ट्रक्शन में मैथ फ्लैग का उपयोग OR इन्स्ट्रक्शन के विभिन्न ऑपरेशनों को परफॉर्म करने के लिए किया जाता है। OR इन्स्ट्रक्शन के अन्तर्गत आने वाले मैथ फ्लैग तथा उनके फंक्शन नीचे सारणी में दिये गये हैं—

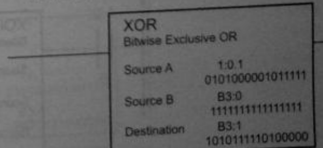
| क्र० सं० | मैथ फ्लैग | स्टेटस | फंक्शन |
|---|---|---|---|
| (i) | कैरी फ्लैग (C) | S : 0/0 | यदि कैरी जनरेट होती है तब यह सेट होता है अन्यथा रीसेट होता है। |
| (ii) | ओवरफ्लो फ्लैग (OV) | S : 0/1 | यदि ओवरफ्लो डेस्टीनेशन में उत्पन्न होता है, तब यह सेट होता है अन्यथा रीसेट होता है। |
| (iii) | जीरो फ्लैग (Z) | S : 0/2 | यदि रिजल्ट जीरो है तब यह सेट होता है अन्यथा रीसेट होता है। |
| (iv) | साइन फ्लैग (S) | S : 0/3 | यदि रिजल्ट निगेटिव है तब यह सेट होता है अन्यथा रीसेट होता है। |

**उदाहरण**—चित्र 3.38 में OR इन्स्ट्रक्शन के उदाहरण को दर्शाया गया है। इसमें सोर्स A को I : 0.1 तथा सोर्स B को B3 : 0 से प्रदर्शित किया गया है। जब इसके द्वारा OR ऑपरेशन परफॉर्म किया जाता है तब यह रिजल्ट को डेस्टीनेशन B3: 1 पर प्रदर्शित करता है। इस प्रकार जब इनपुट I : 0/0 ऑन होता है तब सोर्स A, I : 0.1, 16-बिट तथा सोर्स B3 : 0, 16-बिट OR ऑपरेशन परफॉर्म करता है तथा आउटपुट को डेस्टीनेशन पर स्टोर करता है।




**चित्र 3.38 : लॉजिक OR इन्स्ट्रक्शन का उदाहरण**

**(iii) लॉजिक एक्सक्लूसिव OR (XOR) इन्स्ट्रक्शन** (Logic Exclusive OR (XOR) Instruction)—यह एक आउटपुट इन्स्ट्रक्शन है। जब रंग कन्डीशन सत्य होती है तब यह सोर्स A तथा सोर्स B में एक्सक्लूसिव OR ऑपरेशन को बिट-वाइस परफॉर्म करता है तथा रिजल्ट को डेस्टीनेशन पर स्टोर करता है। एक्सक्लूसिव OR इन्स्ट्रक्शन के फॉर्मेट को चित्र 3.39 में दर्शाया गया है।




**चित्र 3.39 : लॉजिक XOR इन्स्ट्रक्शन का फॉर्मेट**

XOR इन्स्ट्रक्शन के ऑपरेशन में सत्य सारणी का उपयोग XOR इन्स्ट्रक्शन से सम्बन्धित विभिन्न ऑपरेशनों क परफॉर्म करने में किया जाता है। सत्य सारणी में सोर्स A तथा सोर्स B के ऑपरेशन को डेस्टीनेशन द्वारा दर्शाया जा है। XOR इन्स्ट्रक्शन की सत्य सारणी नीचे दी गयी है—

| सोर्स *A* | सोर्स *B* | डेस्टीनेशन |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

XOR इन्स्ट्रक्शन में मैथ फ्लैग का उपयोग XOR इन्स्ट्रक्शन से सम्बन्धित विभिन्न ऑपरेशनों को परफॉर्म कर के लिए किया जाता है। XOR इन्स्ट्रक्शन के अन्तर्गत आने वाले मैथ फ्लैग तथा उनके फंक्शन नीचे सारणी में दिये ग हैं—

| क्र० सं० | मैथ फ्लैग | स्टेटस | फंक्शन |
|---|---|---|---|
| (i) | कैरी फ्लैग (C) | S : 0/0 | यदि कैरी जनरेट होती है तब यह सेट रहता है अन्यथ रीसेट होता है। |
| (ii) | ओवरफ्लो फ्लैग (OV) | S : 0/1 | यदि ओवरफ्लो उत्पन्न होता है, तब यह सेट होता अन्यथा रीसेट होता है। |
| (iii) | जीरो फ्लैग (Z) | S : 0/2 | यदि रिजल्ट जीरो है तब यह सेट होता है अन्यथा रीसे होता है। |
| (iv) | साइन फ्लैग (S) | S : 0/3 | यदि रिजल्ट निगेटिव है तब यह सेट होता है अन्यथा रीसे होता है। |

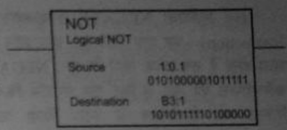
**उदाहरण**—चित्र 3.40 में XOR इन्स्ट्रक्शन के उदाहरण को दर्शाया गया है। इसमें सोर्स A को I : 0.1 तथा सो B को B3 : 0 से प्रदर्शित किया गया है। जब इस इन्स्ट्रक्शन के द्वारा XOR ऑपरेशन परफॉर्म किया जाता है तब य रिजल्ट को डेस्टीनेशन B3 : 1 पर प्रदर्शित किया जाता है। इस प्रकार जब इनपुट I : 0/0 ऑन होता है तब सोर्स A, I 0.1, 16-बिट तथा सोर्स B, B3 : 0, 16-बिट की सहायता से XOR ऑपरेशन परफॉर्म करता है तथा आउटपुट क डेस्टीनेशन पर स्टोर करता है।




**चित्र 3.40 : लॉजिक XOR इन्स्ट्रक्शन का उदाहरण**



**(iv) लॉजिकल NOT इन्स्ट्रक्शन** (Logical NOT Instruction)—यह एक आउटपुट इन्स्ट्रक्शन है। जब रंग कन्डीशन सत्य होती है तब इसके सोर्स द्वारा NOT बिट से बिट तक ऑपरेशन परफॉर्म किया जाता है तथा इसके रिजल्ट को डेस्टीनेशन में स्टोर किया जाता है। NOT इन्स्ट्रक्शन के फॉर्मेट को चित्र 3.41 में दर्शाया गया है।




**चित्र 3.41 : लॉजिकल NOT इन्स्ट्रक्शन का फॉर्मेट**

NOT इन्स्ट्रक्शन में सत्य सारणी का उपयोग NOT इन्स्ट्रक्शन से सम्बन्धित विभिन्न ऑपरेशनों को परफॉर्म करने में किया जाता है। सत्य सारणी के सोर्स A तथा सोर्स B में NOT ऑपरेशन का उपयोग करके उसके डेस्टीनेशन पर दर्शाया जाता है। NOT इन्स्ट्रक्शन की सत्य सारणी नीचे दी गयी है—

| सोर्स | डेस्टीनेशन |
|---|---|
| 0 | 1 |
| 1 | 0 |

NOT इन्स्ट्रक्शन में मैथ फ्लैग का उपयोग विभिन्न NOT ऑपरेशन परफॉर्म करने में किया जाता है। NOT इन्स्ट्रक्शन के अन्तर्गत आने वाले मैथ फ्लैग तथा उनके फंक्शन नीचे सारणी में दिये गये हैं—

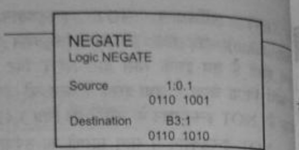
| क्र० सं० | मैथ फ्लैग | स्टेटस | फंक्शन |
|---|---|---|---|
| (i) | कैरी फ्लैग (C) | S : 0/0 | यदि कैरी उत्पन्न होती है तब यह सेट होता है अन्यथा रीसेट होता है। |
| (ii) | ओवरफ्लो फ्लैग (OV) | S : 0/1 | यदि ओवरफ्लो डेस्टीनेशन में उत्पन्न होता है, तब यह सेट होता है अन्यथा रीसेट होता है। |
| (iii) | जीरो फ्लैग (Z) | S : 0/2 | यदि रिजल्ट जीरो है तब यह सेट होता है अन्यथा रीसेट होता है। |
| (iv) | साइन फ्लैग (S) | S : 0/3 | यदि रिजल्ट निगेटिव है तब यह सेट होता है अन्यथा रीसे होता है। |

**उदाहरण**—चित्र 3.42 में NOT इन्स्ट्रक्शन के उदाहरण को दर्शाया गया है। इसमें सोर्स को I : 0.1 त डेस्टीनेशन को B3 : 1 पर प्रदर्शित किया गया है। इस प्रकार जब इसका इनपुट I : 0/0 ऑन होता है तब सोर्स के द्वा एक निश्चित NOT ऑपरेशन परफॉर्म किया जाता है तथा रिजल्ट को डेस्टीनेशन पर स्टोर किया जाता है।




**चित्र 3.42 : NOT इन्स्ट्रक्शन का उदाहरण**

**(v) लॉजिक NEGATE इन्स्ट्रक्शन** (Logic NEGATE Instruction)—यह एक आउटपुट इन्स्ट्रक्शन है। जब रंग कन्डीशन सत्य होती है तब इसके सोर्स के द्वारा NEGATE ऑपरेशन के 2's कॉम्प्लीमेन्ट को बिट से बिट तक परफॉर्म किया जाता है तथा इसके रिजल्ट को डेस्टीनेशन में स्टोर किया जाता है। NEGATE इन्स्ट्रक्शन के फॉर्मेट को चित्र 3.43 में दर्शाया गया है।




**चित्र 3.43 : लॉजिक NEGATE इन्स्ट्रक्शन का फॉर्मेट**

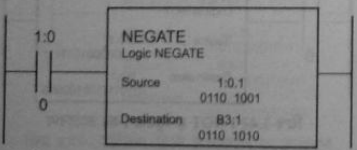
NEGATE इन्स्ट्रक्शन में सत्य सारणी का उपयोग 2's कॉम्प्लीमेन्ट ज्ञात करने के लिए किया जाता है। NEGATE इन्स्ट्रक्शन की सत्य सारणी नीचे दर्शायी गयी है—

| सोर्स | डेस्टीनेशन |
|---|---|
| 0 | 1 |
| 1 | 0 |
| 1 | 1 |
| 0 | 0 |

NEGATE इन्स्ट्रक्शन में मैथ फ्लैग का उपयोग विभिन्न ऑपरेशनों को परफॉर्म करने में किया जाता है। NEGATE इन्स्ट्रक्शन के अन्तर्गत आने वाले मैथ फ्लैग तथा उनके फंक्शन नीचे सारणी में दिये गये हैं—

| क्र० सं० | मैथ फ्लैग | स्टेटस | फंक्शन |
|---|---|---|---|
| (i) | कैरी फ्लैग (C) | S : 0/0 | यदि कैरी उत्पन्न होती है तब यह सेट होता है अन्यथा रीसेट होता है। |
| (ii) | ओवरफ्लो फ्लैग (OV) | S : 0/1 | यदि ओवरफ्लो डेस्टीनेशन में उत्पन्न होता है, तब यह सेट होता है अन्यथा रीसेट होता है। |
| (iii) | जीरो फ्लैग (Z) | S : 0/2 | यदि रिजल्ट जीरो है तब यह सेट होता है अन्यथा रीसेट होता है। |
| (iv) | साइन फ्लैग (S) | S : 0/3 | यदि रिजल्ट निगेटिव है तब यह सेट होता है अन्यथा रीसेट होता है। |

**उदाहरण**—चित्र 3.44 में NEGATE इन्स्ट्रक्शन के उदाहरण को दर्शाया गया है। इसमें सोर्स को I : 0.1 तथा डेस्टीनेशन को B3 : 1 पर प्रदर्शित किया गया है। इस प्रकार जब इसका इनपुट I : 0/0 ऑन होता है तब सोर्स के द्वारा एक निश्चित NEGATE ऑपरेशन परफॉर्म किया जाता है तथा इसके रिजल्ट को डेस्टीनेशन पर स्टोर किया जाता है।



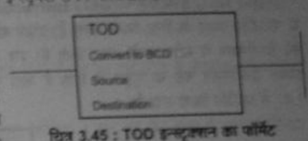

**चित्र 3.44 : NEGATE इन्स्ट्रक्शन का उदाहरण**



## डाटा हैण्डलिंग इन्स्ट्रक्शन (Data Handling Instruction)

डाटा हैण्डलिंग इन्स्ट्रक्शन का उपयोग विभिन्न प्रकार के डाटा हैण्डलिंग से सम्बन्धित ऑपरेशनों को परफॉर्म करने में किया जाता है। डाटा हैण्डलिंग के अन्तर्गत आने वाले प्रमुख सारणी में इन्स्ट्रक्शन्स निम्नलिखित हैं—

**(i) TOD इन्स्ट्रक्शन** (TOD Instruction)—TOD इन्स्ट्रक्शन, किसी संख्या को BCD में कन्वर्ट करता है। इस इन्स्ट्रक्शन के द्वारा इन्टीजर सोर्स को BCD फॉर्मेट में बदला जाता है तथा रिजल्ट को डेस्टीनेशन में स्टोर करके रखा जाता है। इस प्रकार से इस इन्स्ट्रक्शन में सोर्स वर्ड एड्रेस या कॉन्स्टेन्ट के रूप में हो सकता है तथा डेस्टीनेशन वर्ड एड्रेस या रजिस्टर पर प्राप्त होता है। चित्र 3.45 में TOD इन्स्ट्रक्शन के फॉर्मेट को दर्शाया गया है।

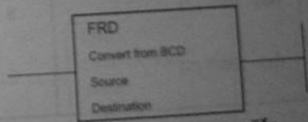



**चित्र 3.45 : TOD इन्स्ट्रक्शन का फॉर्मेट**

TOD इन्स्ट्रक्शन में अर्थमैटिक स्टेटस बिट का उपयोग TOD इन्स्ट्रक्शन से सम्बन्धित विभिन्न ऑपरेशनों को परफॉर्म करने में किया जाता है। TOD इन्स्ट्रक्शन के अन्तर्गत आने वाले अर्थमैटिक स्टेटस बिट तथा उनके फंक्शन नीचे सारणी में दिये गये हैं—

| क्र० सं० | अर्थमैटिक स्टेटस बिट | स्टेटस | फंक्शन |
|---|---|---|---|
| (i) | कैरी फ्लैग (C) | S : 0/0 | इसमें कैरी फ्लैग सेट रहता है। |
| (ii) | ओवरफ्लो (OV) | S : 0/1 | यदि BCD रिजल्ट 9999 से बड़ा है तब यह सेट होता है। इसके ओवरफ्लो रिजल्ट में माइनस एरर होती है। |
| (iii) | जीरो (Z) | S : 0/2 | यदि डेस्टीनेशन वेल्यू जीरो है तब यह सेट होता है। |
| (iv) | साइन (S) | S : 0/3 | यदि सोर्स, वर्ड इन्टीजर में होता है तब यह सेट होता है अन्यथा रीसेट होता है। |

**(ii) FRD इन्स्ट्रक्शन** (FRD Instruction)—FRD इन्स्ट्रक्शन का उपयोग किसी संख्या को BCD से बाइनरी, डेसीमल तथा हेक्साडेसीमल आदि में परिवर्तित करने में किया जाता है। इसमें BCD सोर्स वेल्यू को इन्टीजर में कन्वर्ट किया जाता है तथा रिजल्ट को डेस्टीनेशन में स्टोर करके रखा जाता है। इस ऑपरेशन में सोर्स या तो वर्ड एड्रेस में होता है या कॉन्स्टेन्ट में होता है। इसका डेस्टीनेशन वर्ड एड्रेस या रजिस्टर में होता है। FRD इन्स्ट्रक्शन के फॉर्मेट को चित्र 3.46 में दर्शाया गया है।



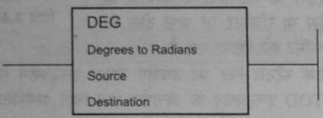

**चित्र 3.46 : FRD इन्स्ट्रक्शन का फॉर्मेट**

FRD इन्स्ट्रक्शन में अर्थमैटिक स्टेटस बिट का उपयोग FRD इन्स्ट्रक्शन से सम्बन्धित विभिन्न ऑपरेशनों को परफॉर्म करने में किया जाता है। FRD इन्स्ट्रक्शन के अन्तर्गत आने वाले अर्थमैटिक स्टेटस बिट तथा उनके फंक्शन नीचे सारणी में दिये गये हैं—

| क्र० सं० | अर्थमैटिक स्टेटस बिट | स्टेटस | फंक्शन |
|---|---|---|---|
| (i) | कैरी फ्लैग (C) | S:0/0 | इसमें कैरी फ्लैग रीसेट रहता है। |
| (ii) | ओवरफ्लो (OV) | S:0/1 | इसमें नॉन-BCD वेल्यू को रखा जाता है या जो वेल्यू कन्वर्ट हुई है वह 32, 761 से बड़ी होनी चाहिए तब यह सेट होता है अन्यथा रीसेट होता है। |

| | | | |
|---|---|---|---|
| (iii) | जीरो (Z) | S:0/2 | यदि डेस्टीनेशन वेल्यू जीरो व तब यह सेट होता है। |
| (iv) | साइन (S) | S:0/3 | साइन फ्लैग रीसेट रहता है। |

**(iii) DEG इन्स्ट्रक्शन (DEG Instruction)**—DEG इन्स्ट्रक्शन का उपयोग किसी वेल्यू को डिग्री से रेडि में परिवर्तित करने के लिए किया जाता है। इसमें सोर्स वेल्यू (डिग्री) को रेडियन में परिवर्तित किया जाता है तथा रिज को डेस्टीनेशन में स्टोर करके रखा जाता है। इस ऑपरेशन में सोर्स या तो इन्टीजर में होता है या फ्लोटिंग में होता इसका डेस्टीनेशन वर्ड या एड्रेस में होता है जहाँ पर डाटा स्टोर किया जाता है। DEG इन्स्ट्रक्शन के फॉर्मेट को चि 3.47 में प्रदर्शित किया गया है।

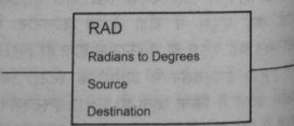



**चित्र 3.47 : DEG इन्स्ट्रक्शन का फॉर्मेट**

DEG इन्स्ट्रक्शन में अर्थमैटिक स्टेटस बिट का उपयोग DEG इन्स्ट्रक्शन से सम्बन्धित विभिन्न ऑपरेशनों परफॉर्म करने में किया जाता है। DEG इन्स्ट्रक्शन के अन्तर्गत आने वाले अर्थमैटिक स्टेटस बिट तथा उनके फंक्शन नी सारणी में दिये गये हैं—

| क्र० सं० | अर्थमैटिक स्टेटस बिट | स्टेटस | फंक्शन |
|---|---|---|---|
| (i) | कैरी फ्लैग (C) | S : 0/0 | इसमें कैरी फ्लैग सेट रहता है। |
| (ii) | ओवरफ्लो (OV) | S : 0/1 | यदि ओवरफ्लो जनरेट होता है या अनसपोर्ट इनपुट डिटे होता है तब यह सेट होता है अन्यथा रीसेट होता है। |
| (iii) | जीरो (Z) | S : 0/2 | यदि रिजल्ट जीरो है तब यह सेट होता है अन्यथा रीसेट हो है। |
| (iv) | साइन (S) | S : 0/3 | यदि रिजल्ट निगेटिव है तब यह सेट होता है अन्यथा रीसे होता है। |

**(iv) RAD इन्स्ट्रक्शन** (RAD Instruction)—RAD इन्स्ट्रक्शन का उपयोग किसी वेल्यू को रेडियन से डिग्री में परिवर्तित करने के लिए किया जाता है। इसमें सोर्स वेल्यू (रेडियन) को डिग्री में परिवर्तित किया जाता है तथा रिजल्ट को डेस्टीनेशन में स्टोर करके रखा जाता है। इस ऑपरेशन में सोर्स या तो इन्टीजर फॉर्म में होता है या फ्लोटिंग फॉर्म में होता है। इसका डेस्टीनेशन, वर्ड या एड्रेस में होता है जहाँ पर डाटा स्टोर करके रखा जाता है। RAD इन्स्ट्रक्शन के फॉर्मेट को चित्र 3.48 में प्रदर्शित किया गया है।



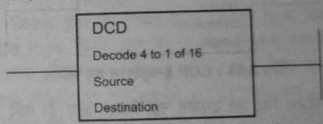

**चित्र 3.48 : RAD इन्स्ट्रक्शन का फॉर्मेट**

RAD इन्स्ट्रक्शन में अर्थमैटिक स्टेटस बिट का उपयोग RAD इन्स्ट्रक्शन से सम्बन्धित ऑपरेशनों को परफॉ करने में किया जाता है। RAD इन्स्ट्रक्शन के अन्तर्गत आने वाले अर्थमैटिक स्टेटस बिट तथा उनके इन्स्ट्रक्शन नी सारणी में दिये गये हैं—



| क्र० सं० | अर्थमैटिक स्टेटस बिट | स्टेटस | फंक्शन |
|---|---|---|---|
| (i) | कैरी फ्लैग (C) | S : 0/0 | इसमें कैरी फ्लैग रीसेट रहता है। |
| (ii) | ओवरफ्लो (OV) | S : 0/1 | इसमें ओवरफ्लो जनरेट होता है या अनसपोर्ट इनपुट डिटेक्ट होता है तब यह सेट होता है अन्यथा रीसेट होता है। |
| (iii) | जीरो (Z) | S : 0/2 | यदि रिजल्ट जीरो है तब यह सेट होता है अन्यथा रीसेट होता है। |
| (iv) | साइन (S) | S : 0/3 | यदि रिजल्ट निगेटिव है तब यह सेट होता है अन्यथा रीसेट होता है। |

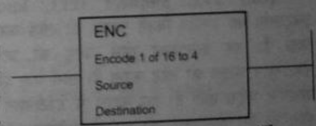
**(v) DCD इन्स्ट्रक्शन** (DCD Instruction)—DCD इन्स्ट्रक्शन का उपयोग किसी वेल्यू को डिकोड करने में किया जाता है। इसमें 4-बिट वेल्यू (0 से 15) को उससे सम्बन्धित 16-बिट के डेस्टीनेशन पर भेजा जाता है। इसमें सोर्स एक एड्रेस होता है जो बिट डिकोड की इन्फॉर्मेशन को रखता है। इसमें केवल चार बिट (0 से 3) को DCD इन्स्ट्रक्शन में उपयोग किया जाता है तथा शेष बिट को अन्य स्पेसीफिक आवश्यकताओं में उपयोग किया जाता है। इसमें प्रथम चार बिट को इस वर्ड की सहायता से बदला जाता है तथा डेस्टीनेशन में इस एक बिट को प्रदर्शित किया जाता है। DCD इन्स्ट्रक्शन के फॉर्मेट को चित्र 3.49 में दर्शाया गया है।




**चित्र 3.49 : DCD इन्स्ट्रक्शन का फॉर्मेट**

DCD इन्स्ट्रक्शन में अर्थमैटिक स्टेटस बिटें अप्रभावित रहती हैं।

**(vi) ENC इन्स्ट्रक्शन** (ENC Instruction)—ENC इन्स्ट्रक्शन का उपयोग किसी वेल्यू का एनकोड ज्ञात करने में किया जाता है। इस प्रकार सोर्स में एनकोर्ड 16-बिट से 4-बिट तक होता है। इसमें सोर्स को लोअर बिट से हायर बिट तक प्रथम सेट बिट की सहायता से रीसर्च किया जाता है। ENC इन्स्ट्रक्शन से सम्बन्धित बिट की पोजीशन को डेस्टीनेशन में इन्टीजर की तरह लिखा जाता है। इसमें, वर्ड सोर्स का एड्रेस होता है जिसे डिकोड किया जाता है। इस वर्ड में केवल एक बिट कभी भी ऑन हो सकती है। यदि एक से अधिक बिट सोर्स में है तब डेस्टीनेशन बिट लीस्ट सिग्नीफिकेन्ट बिट के आधार पर सेट होती है। इसमें जब जीरो सोर्स का उपयोग किया जाता है तब सभी डेस्टीनेशन बिटें सीरेट होती हैं तथा जीरो बिट सेट होती है। इसमें डेस्टीनेशन एक एड्रेस होता है जो एनकोड से सम्बन्धित इन्फॉर्मेशन को रखता है। ENC इन्स्ट्रक्शन के द्वारा 4 से 15 बिट को डेस्टीनेशन में सेट किया जाता है। ENC इन्स्ट्रक्शन के फॉर्मेट को चित्र 3.50 में दर्शाया गया है।

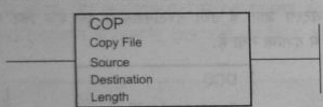



**चित्र 3.50 : ENC इन्स्ट्रक्शन का फॉर्मेट**

ENC इन्स्ट्रक्शन में अर्थमैटिक स्टेटस बिट का उपयोग ENC इन्स्ट्रक्शन से सम्बन्धित ऑपरेशनों को परफॉर्म करने में किया जाता है। ENC इन्स्ट्रक्शन के अन्तर्गत आने वाले अर्थमैटिक स्टेटस बिट तथा उनके फंक्शन नीचे सारणी में दिये गये हैं—

| क्र० सं० | अर्थमैटिक स्टेटस बिट | स्टेटस | फंक्शन |
|---|---|---|---|
| (i) | कैरी फ्लैग (C) | S : 0/0 | इसमें कैरी फ्लैग रीसेट होता है। |
| (ii) | ओवरफ्लो (OV) | S : 0/1 | यदि एक से अधिक बिटें सोर्स में हैं तो यह सेट होता अन्यथा रीसेट होता है। इसमें मैथ ओवरफ्लो बिट सेट होता है। |
| (iii) | जीरो (Z) | S : 0/2 | यदि डेस्टीनेशन वेल्यू जीरों है तब यह सेट होता है। |
| (iv) | साइन (S) | S : 0/3 | इसमें साइन फ्लैग रीसेट रहता है। |

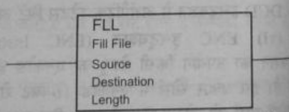
**(vii) COP इन्स्ट्रक्शन** (COP Instruction)—COP इन्स्ट्रक्शन का उपयोग किसी फाइल को एक लोकेशन दूसरे लोकेशन मे कॉपी करने के लिए किया जाता है। इस इन्स्ट्रक्शन की सहायता से सोर्स फाइल से डाटा डेस्टीनेशन फाइल में कॉपी किया जाता है। इसमें सोर्स एक फाइल एड्रेस होता है तथा डेस्टीनेशन एक स्ट्रिंग एड्रेस है जहाँ पर इन्स्ट्रक्शनों को कॉपी तथा स्टोर किया जाता है। इस इन्स्ट्रक्शन की लेन्थ फाइल के नम्बर ऑफ एलीमें पर निर्भर करती है। COP इन्स्ट्रक्शन के फॉर्मेट को चित्र 3.51 में प्रदर्शित किया गया है।




**चित्र 3.51 : COP इन्स्ट्रक्शन का फॉर्मेट**

इस इन्स्ट्रक्शन में अर्थमैटिक स्टेटस बिट का उपयोग नहीं किया जाता है। यदि इसमें इनेबल बिट की आवश्य होती है तब इसे आउटपुट इन्स्ट्रक्शन (OTE) की सहायता से इन्टर्नल बिट के आउटपुट के साथ प्रोग्रामेबल बनाया है।

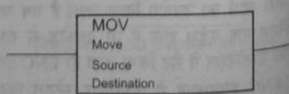
**(viii) FLL इन्स्ट्रक्शन** (FLL Instruction)—FLL इन्स्ट्रक्शन का उपयोग किसी फाइल को लोड करने के लिए किया जाता है। यह इन्स्ट्रक्शन सोर्स वेल्यू की प्रत्येक पोजीशन में डेस्टीनेशन फाइल को लोड करता है। सोर्स एक प्रोग्राम कॉन्स्टेन्ट या एलीमेन्ट एड्रेस होता है। इस फाइल में डेस्टीनेशन एक स्ट्रिंग एड्रेस होता है। इस इन्स्ट्रक्शन की लम्बाई एलीमेन्ट की संख्या पर निर्भर करती है। FLL इन्स्ट्रक्शन के फॉर्मेट को चित्र 3.52 में प्रदर्शित किया गया है।




**चित्र 3.52 : FLL इन्स्ट्रक्शन का फॉर्मेट**

इस इन्स्ट्रक्शन के अर्थमैटिक स्टेटस बिट 0 वर्ड में पायी जाती है तथा कन्ट्रोलर स्टेटस फाइल 0 से 3 बिट होती है।

**(ix) MOV इन्स्ट्रक्शन** (MOV Instruction)—MOV इन्स्ट्रक्शन का उपयोग किसी फाइल की एक लोकेशन से दूसरे लोकेशन में मूल कराने के लिए किया जाता है। इस इन्स्ट्रक्शन में सोर्स वेल्यू को डेस्टीनेशन में मूव किया जाता है। इसमें सोर्स, डाटा का एड्रेस या कॉन्स्टेन्ट होता है जिसे मूव कराया जाता है। डेस्टीनेशन वह एड्रेस होता है जहाँ इन्स्ट्रक्शन डाटा को मूव करता है। MOV इन्स्ट्रक्शन के फॉर्मेट को चित्र 3.53 में प्रदर्शित कि गया है। यह आउटपुट इन्स्ट्रक्शन सोर्स वेल्यू को डेस्टीनेशन लोकेशन पर मूव कराता है। इस इन्स्ट्रक्शन में जैसे ही कन्डीशन सत्य होती है वैसे ही इन्स्ट्रक्शन प्रत्येक स्कैन में डाटा को मूव करता है।



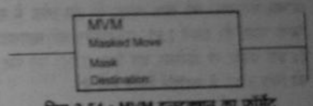

**चित्र 3.53 : MOV इन्स्ट्रक्शन का फॉर्मेट**



MOV इन्स्ट्रक्शन में अर्थमैटिक स्टेटस बिट का उपयोग MOV इन्स्ट्रक्शन से सम्बन्धित विभिन्न ऑपरेशनों को रफॉर्म करने में किया जाता है। MOV इन्स्ट्रक्शन के अन्तर्गत आने वाले अर्थमैटिक स्टेटस तथा उनके फंक्शन नीचे सारणी में दिये गये है—

| क्र० सं० | अर्थमैटिक स्टेटस बिट | स्टेटस | फंक्शन |
|---|---|---|---|
| (i) | कैरी फ्लैग (C) | S : 0/0 | इसमें कैरी हमेशा रीसेट होती है। |
| (ii) | ओवरफ्लो (OV) | S : 0/1 | इसमें ओवरफ्लो हमेशा रीसेट रहता है। |
| (iii) | जीरो (Z) | S : 0/2 | यदि रिजल्ट जीरो है तब यह सेट होता है अन्यथा रीसेट होता है। |
| (iv) | साइन (S) | S : 0/3 | यदि रिजल्ट निगेटिव होता है तब यह सेट होती है अन्यथा रीसेट होता है। |

**(x) MVM इन्स्ट्रक्शन** (MVM Instruction)—MVM न्स्ट्रक्शन का उपयोग किसी वेल्यू को मास्क में बदलकर एक लोकेशन से दूसरे लोकेशन में मूव किया जाता है। इस इन्स्ट्रक्शन में सोर्स लोकेशन के डाटा को डेस्टीनेशन के सिलेक्ट पोर्शन में मूव किया जाता है। सोर्स, एड्रेस का डाटा होता है जिसे मूव किया जाता है। मास्क का एड्रेस मास्क होता है जिसके द्वारा इन्स्ट्रक्शन डाटा को मूव किया जाता है। मास्क हेक्साडेसीमल वेल्यू होती है जिसे एक निश्चित एड्रेस द्वारा मास्क किया जाता है। इसमें डेस्टीनेशन एक एड्रेस होता है जहाँ पर इन्स्ट्रक्शन डाटा को मूव करता है। MVM इन्स्ट्रक्शन के फॉर्मेट को चित्र 3.54 में प्रदर्शित किया गया है।



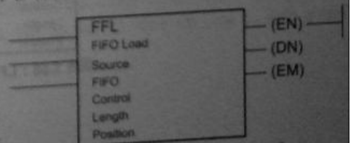

**चित्र 3.54 : MVM इन्स्ट्रक्शन का फॉर्मेट**

MVM इन्स्ट्रक्शन में अर्थमैटिक स्टेटस बिट का उपयोग MVM इन्स्ट्रक्शन से सम्बन्धित विभिन्न ऑपरेशनों को रफॉर्म करने में किया जाता है। MVM इन्स्ट्रक्शन के अन्तर्गत आने वाले अर्थमैटिक स्टेटस बिट तथा उनके फंक्शन नीचे सारणी में दिये गये हैं—

| क्र० सं० | अर्थमैटिक स्टेटस बिट | स्टेटस | फंक्शन |
|---|---|---|---|
| (i) | कैरी फ्लैग (C) | S : 0/0 | इसमें कैरी हमेशा रीसेट होती है। |
| (ii) | ओवरफ्लो (OV) | S : 0/1 | इसमें ओवरफ्लो हमेशा रीसेट होता है। |
| (iii) | जीरो (Z) | S : 0/2 | यदि रिजल्ट जीरो है तब यह सेट होता है अन्यथा रीसेट होता है। |
| (iv) | साइन (S) | S : 0/3 | यदि रिजल्ट निगेटिव है तब यह सेट होता है अन्यथा रीसेट होता है। |

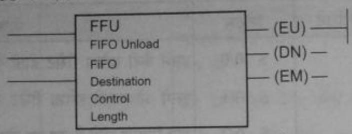
**(xi) FFL इन्स्ट्रक्शन या FIFO लोड** (FFL Instruction or FIFO Load)— FFL इन्स्ट्रक्शन में फर्स्ट इन फर्स्ट आउट के आधार पर डाटा को लोड किया जाता है। FFL इन्स्ट्रक्शन वर्ड को यूजर द्वारा उत्पन्न की गयी फाइल में लोड करता है जिसे FIFO स्टैक कहा जाता है। FFL इन्स्ट्रक्शन के फॉर्मेट को चित्र 3.55 में प्रदर्शित किया गया है।

**(xii) FFU इन्स्ट्रक्शन या FIFO अनलोड** (FFU Instruction or FIFO Unload)—FFL तथा FFU इन्स्ट्रक्शन का उपयोग साथ-साथ किया जाता है। FFU इन्स्ट्रक्शन का उपयोग फर्स्ट इन फर्स्ट आउट के आधार पर किसी फाइल में डाटा को लोड तथा अनलोड करने में किया



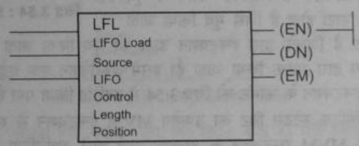

**चित्र 3.55 : FFL इन्स्ट्रक्शन का फॉर्मेट**

जाता है। FFL इन्स्ट्रक्शन यूजर द्वारा उत्पन्न फाइल के वर्ड को लोड करता है जिसे FIFO स्टैक कहा जाता है। FFU इन्स्ट्रक्शन वर्ड को FIFO स्टैक से अनलोड उसी क्रम में करता है जिस क्रम में लोड किया गया था। इन्स्ट्रक्शन के फॉर्मेट को चित्र 3.56 में प्रदर्शित किया गया है।



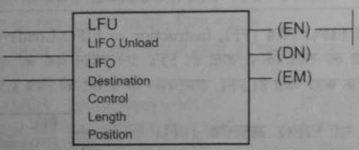

चित्र 3.56 : FFU इन्स्ट्रक्शन का फॉर्मेट

**(xiii) LFL इन्स्ट्रक्शन या LIFO लोड** (LFL Instruction or LIFO Load)—जब रंग कन्डीशन अस सत्य में बदलती है तब LFL एम्पटी बिट (EM) सेट होती है। LFL इन्स्ट्रक्शन लास्ट इन फर्स्ट आउट (LIF आधार पर डाटा को लोड करता है। इस लोड में सोर्स N7 : 10 होता है जिसमें स्टैक एलीमेन्ट को 9 नम्बर से इ किया जाता है। इसमें LFL इन्स्ट्रक्शन की सहायता से रंग प्रत्येक एलीमेन्ट को असत्य से सत्य ट्रांजीशन में लोड है। इस प्रकार से प्रोसेसर डन बिट (DN) को सेट करता है इसके बाद उसे लोड करता है। LFL इन्स्ट्रक्शन के को चित्र 3.57 में प्रदर्शित किया गया है।




चित्र 3.57 : LFL इन्स्ट्रक्शन का फॉर्मेट

**(xiv) LFU इन्स्ट्रक्शन या LIFO अनलोड** (LFU Instruction or LIFO Unload)—जब रंग क असत्य से सत्य में बदलती है तब LFU इनेबल बिट (EN) सेट होती है। इस प्रकार से यह अनलोड डाटा को एलीमेन्ट से स्टैक को डेस्टीनेशन N7 : 11 के स्थान पर अनलोड किया जाता है। इसमें पहली पोजीशन वे डिक्रीमेन्ट किया जाता हे। इसमें रंग के प्रत्येक एलीमेन्ट को असत्य से सत्य ट्रांजीशन पर LFU इन्स्ट्रक्शन की स से अनलोड किया जाता है जब तक कि स्टैक खाली रहता है। इसमें प्रोसेसर एम्पटी बिट (EM) पर सेट होता है। इन्स्ट्रक्शन के फॉर्मेट को चित्र 3.58 में प्रदर्शित किया गया है।




चित्र 3.58 : LFU इन्स्ट्रक्शन का फॉर्मेट



## PID कन्ट्रोल, PID इक्वेशन, PID इन्स्ट्रक्शन (PID Control, PID Equation, PID Instruction)

**(i) PID कन्ट्रोल** (PID Control)—इन्डस्ट्रीज में विभिन्न प्रकार की कन्ट्रोल टेक्निक प्रक्रियाएँ उपयोग की जाती । प्रपोशनल इन्टीग्रल डेरीवेटिव कन्ट्रोल एक बहुत ही महत्वपूर्ण टेक्निक है जो इन्डस्ट्रीज में उपयोग की जाती है। इस क्निक का टेक्नीकल उपयोग अधिक कठिन है तथा इसकी रेन्ज प्राप्त करना आसान है। PID कन्ट्रोल, कन्ट्रोल सिस्टम ी टेक्निक की सहायता से सरल एवं आसान बनाया जाता है। इसके कन्ट्रोल के लिए क्लोज लूप इनपुट का उपयोग नालॉग इनपुट मॉड्यूल के रूप में किया जाता है तथा इसका आउटपुट एनालॉग, आउटपुट मॉड्यूल के रूप में प्राप्त ोता है।

**(ii) PID इक्वेशन** (PID Equation)—PID इक्वेशन कन्ट्रोल वेल्यू आउटपुट सिग्नल भेजकर प्रोसेस को न्ट्रोल करती है। इसमें एरर सेट पॉइन्ट तथा प्रोसेस वेरियेबल के बीच उत्पन्न होती है, जिसमें आउटपुट सिग्नल नालॉग होता है। इसके कन्ट्रोल आउटपुट में एडीशनल वेल्यू (फीड फॉरवर्ड) को ऑफसेट के रूप में जोड़ा जाता है। समें PID के रिजल्ट को कैलकुलेशन की सहायता से प्रोसेस वेरियबल से पहले सेट पॉइन्ट पर प्राप्त किया जाता है। स प्रकार से कन्ट्रोल स्टेण्डर्ड इक्वेशन डिपेन्डेन्ट गेन आधारित होता है जो निम्नलिखित है—

$$\text{आउटपुट} = K_c\,[(E) + \frac{1}{T_1}\int (E)\,dt + T_D \cdot D(PV)/df] + bais$$

जहाँ पर—

$K_c$ = कन्ट्रोलर गेन,

$\frac{1}{T_1}$ = रीसेट टर्म,

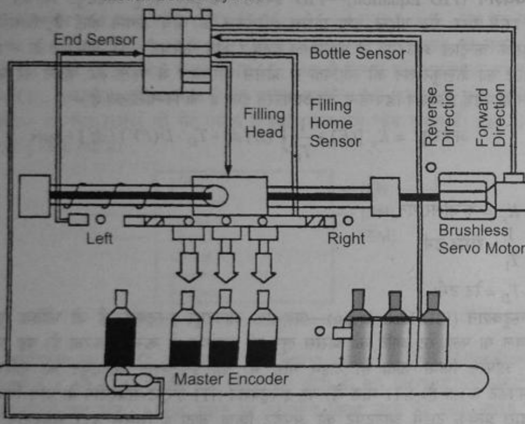
$T_D$ = रेट टर्म।

**(iii) PID इन्स्ट्रक्शन** (PID Instruction)—यह एक आउटपुट इन्स्ट्रक्शन है जो भौतिक गुणों, जैसे—टेम्प्रेचर, शर, लिक्विड, लेवल या फ्लो रेट आदि को प्रोसेस लूप की सहायता से कन्ट्रोल करता है। यह इन्स्ट्रक्शन टाइम मोड या STI मोड पर ऑपरेट किया जाता है। टाइम मोड में, यह इन्स्ट्रक्शन आउटपुट को पीरियोडिकली यूजर की आवश्यकतानुसार अपडेट करता है। STI मोड में, यह इन्स्ट्रक्शन STI इन्ट्रप्ट सबरुटीन के द्वारा निर्धारित किया जाता है। इस प्रकार इसके द्वारा प्रत्येक टाइम आउटपुट को अपडेट किया जाता है जिससे STI सबरुटीन स्कैन करता है। इस प्रकार से जब STI टाइम इन्टर्वल तथा PID लूप सही क्रम में अपडेट होता है तब इसकी इक्वेशन को सही क्रम में एक्जीक्यूट किया जाता है।

प्रोग्रामिंग के दौरान, कन्ट्रोल ब्लॉक में प्रोसेस वेरियेबल तथा कन्ट्रोल वेरियेबल एड्रेस एन्टर किये जाते हैं। कन्ट्रोल ब्लॉक एक फाइल है जो डाटा में आवश्यक इन्स्ट्रक्शन को ऑपरेट करने के लिए स्टोर करता है। इसमें प्रोसेस वेरियेबल एक एलीमेन्ट एड्रेस होता है जो प्रोसेस इनपुट वेल्यू को स्टोर करता है। इस एड्रेस को एनालॉग इनपुट वर्ड के रूप में लोकेट किया जाता है, जहाँ पर इनपुट की वेल्यू स्टोर रहती है। कन्ट्रोल वेरियेबल एक एलीमेन्ट एड्रेस है जो PID इन्स्ट्रक्शन के आउटपुट को स्टोर करता है। इसमें कुछ अन्य पैरामीटर्स, जैसे—ऑटो-मैनुअल AM (वर्ड 0, बिट 1) टाइम मोड TM (वर्ड 0, बिट 0), कन्ट्रोल मोड CM (वर्ड 0, बिट 2) तथा सेट पॉइन्ट SP (वर्ड 2) आदि उपयोग किये जाते हैं।

## ऑटोमेटिड सिस्टम में बोतलिंग लाइन प्रक्रिया का फ्लो डायग्राम तथा लैडर लॉ[जिक] डायग्राम (Flow diagram and Ladder logic diagram of Automatic bottling pla[nt])

बोतलिंग लाइन प्रक्रिया मे ऑटामेटिड सिस्टम का उपयोग किया जाता है। इस प्रकार इसमें कन्वेयर भरी हुई [...] को डेस्टीनेशन पर ट्रांसफर करता है। इस सिस्टम में दो दायें तथा बायें सेन्सर, बोतल फिलिंग प्रक्रिया में उपयो[ग] [...] जाते है। इसमें बोतल एक क्रम के अनुसार भरती है। इसमें सर्वप्रथम लाइन पर खाली बोतलों को लगाया जाता है [...] बाद इन्हें भरा जाता है। चित्र 3.59 में बोतलिंग लाइन प्रक्रिया के फ्लो डायग्राम तथा चित्र 3.60 में बोतलिंग ला[इन] लैडर लॉजिक डायग्राम को दर्शाया गया है।

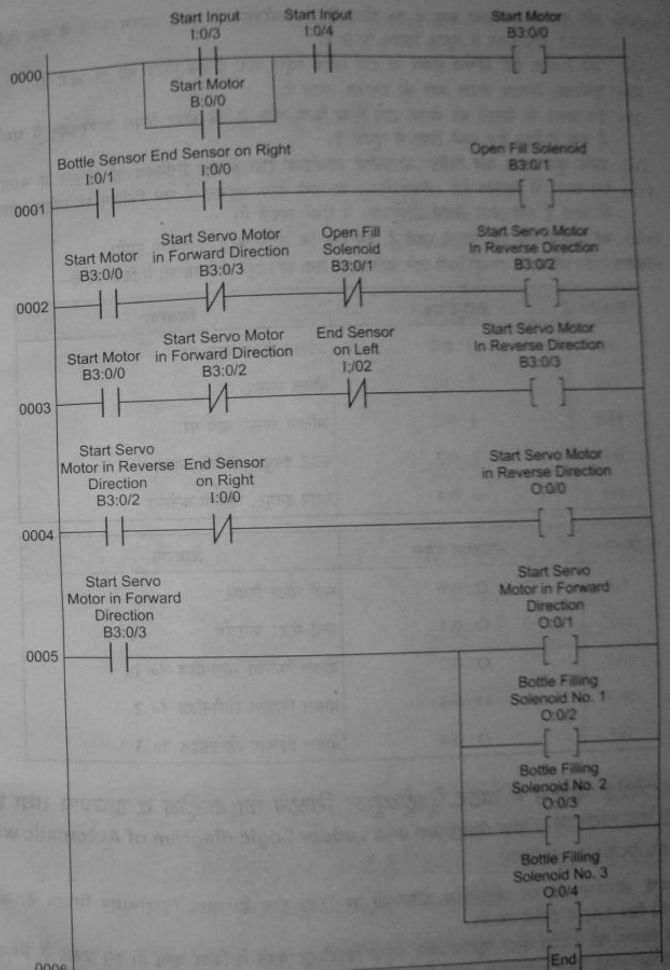



चित्र 3.59 : बोतलिंग लाइन प्रक्रिया का फ्लो डायग्राम

बोतलिंग लाइन सिस्टम के लैडर लॉजिक को इम्प्लीमेन्ट करने के प्रमुख कन्ट्रोलिंग फंक्शन निम्नलिखित हैं—

(i) इसमें फिलिंग हैड दो सेन्सर के बीच होता है।






चित्र 3.60 : बोतलिंग लाइन लैडर लॉजिक डायग्राम

(ii) जब पुश बटन दबाया जाता है तब मोटर रिवर्स डायरेक्शन में घूमना प्रारम्भ करती है तथा फिलि[illegible] फारवर्ड डायरेक्शन में घूमना प्रारम्भ करता है।

(iii) जब फिलिंग हैड अन्तिम सेन्सर या दायें साइड पहुँच जाता है तब मोटर बंद हो जाती है।

(iv) तत्पश्चात् सिस्टम बोतल आने का इन्तजार करता है।

(v) इस प्रकार से बोतलों को सेन्सर द्वारा सेन्स किया जाता है तब मोटर रिवर्स डायरेक्शन में घूमने [illegible] है तथा फिलिंग हैड बायीं दिशा में घूमता है।

(vi) इसके कुछ समय बाद फिलिंग सोलेनॉइड इनर्जाइज्ड होता है तथा लिक्विड को बोतलों में भरता है।

(vii) इस प्रकार से फिलिंग हैड अन्तिम सेन्सर या बायीं तरफ पहुँचता है तब फिलिंग सोलेनॉइड डीइन[illegible] हो जाता है तथा मोटर रिवर्स डायरेक्शन में घूमने लगती है।

(viii) यह प्रक्रिया तब तक चलती रहती है जब तक कि स्टॉप बटन न दबाया जाये।

बोतलिंग लाइन प्रक्रिया में उपयोग किये जाने वाले इनपुट तथा आउटपुट नीचे सारणी में दिये गये हैं—

| क्र०सं० | इनपुट एड्रेस | विवरण |
|---|---|---|
| (i) | I : 0/0 | अन्तिम सेन्सर दायें पर |
| (ii) | I : 0/1 | बोतल सेन्सर |
| (iii) | I : 0/2 | अन्तिम सेन्सर बायें पर |
| (iv) | I : 0/3 | स्टार्ट इनपुट, नॉर्मली ओपन |
| (v) | I : 0/4 | स्टॉप इनपुट, नॉर्मली क्लोज्ड |

| क्र०सं० | आउटपुट एड्रेस | विवरण |
|---|---|---|
| (i) | O : 0/0 | सर्वो मोटर रिवर्स |
| (ii) | O : 0/1 | सर्वो मोटर फारवर्ड |
| (iii) | O : 0/2 | बोतल फिलिंग सोलेनॉइड नं० 1 |
| (iv) | O : 0/3 | बोतल फिलिंग सोलेनॉइड नं० 2 |
| (v) | O : 0/4 | बोतल फिलिंग सोलेनॉइड नं० 3 |

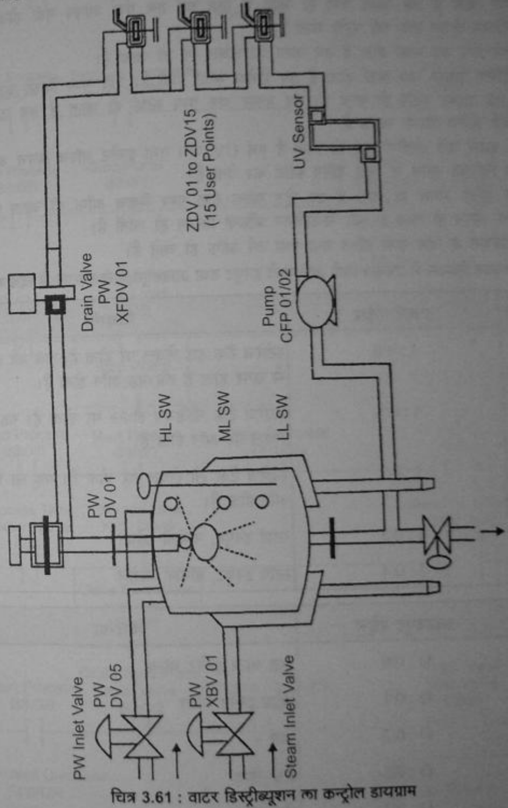
## आटोमेटिड सिस्टम में वाटर ड्रिस्ट्रीब्यूशन सिस्टम का कन्ट्रोल व डायग्राम तथा लै[illegible] लॉजिक डायग्राम (Flow diagram and Ladder Logic diagram of Automatic wa[illegible] distribution system)

सम्पूर्ण ऑटोमेशन सिस्टम ऑटोमेटिक प्रक्रियाओं पर टिका हुआ है। वाटर डिस्ट्रीब्यूशन सिस्टम के क[illegible] डायग्राम को चित्र 3.61 में दर्शाया गया है।

इस सिस्टम का उपयोग वाटर प्यूरीफिकेशन उसको डिस्ट्रीब्यूट करने में किया जाता है। इस प्रकार के सिस्ट[illegible] शुद्ध वाटर को इन्लेट वॉल्व DV 05 से इन्सर्ट किया जाता है, स्ट्रीम इन्लेट के लिए XBV 01, ड्रेन इन्लेट के लिए [illegible] DV 01 का उपयोग किया जाता है। इस प्रकार के सिस्टम में हाई लेवल स्विच को HL SW, मीडियम लेवल स्विच [illegible]



ML SW तथा लो लेवल स्विच को LL SW से दर्शाया गया है। इनका उपयोग वाटर टैंक के लेवल के लिए किया जाता है। इसमें पम्प के लिए CEP 01/02 का उपयोग किया जाता है तथा वाटर को प्यूरीफाई करने के लिए UV सेन्सर का उपयोग किया जाता है।




चित्र 3.61 : वाटर डिस्ट्रीब्यूशन का कन्ट्रोल डायग्राम

इस सिस्टम का उपयोग इन्डस्ट्रीज के विभिन्न ऑपरेशनों को परफॉर्म करने में किया जाता है। चिलर, बॉयलर तथा कटर आदि में इस फंक्शन का उपयोग अधिक किया जाता है। चित्र 3.62 में वाटर डिस्ट्रीब्यूशन के लैडर लॉजिक डायग्राम को दर्शाया गया है।

वाटर डिस्ट्रीब्यूशन सिस्टम के लैडर लॉजिक को इम्प्लीमेन्ट करने के प्रमुख कन्ट्रोलिंग फंक्शन निम्नलिखित है—

(i) जब पुश बटन को दबाते हैं तब UV लेम्प टर्न ऑन हो जाती है।

(ii) शुद्ध वाटर इन्लेट वॉल्व, ओपन होकर वाटर को अन्दर आने देता है। इसमें यदि वाटर हाई लेवल ऊपर जाता है तब वॉल्व बन्द हो जाता है तथा यह तब तक ओपन नहीं होता है जब तक मीडियम लेवल तक नहीं पहुँच जाता है।

(iii) इसमें पम्प जब स्टार्ट होता है तब वाटर का लेवल लो हो जाता है।

(iv) रिटेन्टिव टाइमर जब स्टार्ट होता है तब प्रोसेस स्टार्ट होती है। यदि पम्प किसी कारण से बंद हो है तब टाइमर स्टॉप हो जाता है। इस प्रकार जब पम्प स्टार्ट हो जाता है तब टाइमर उसी स्थिति अपनी प्रोसेस प्रारम्भ करता है।

(v) इस प्रकार यदि टाइमिंग पूर्ण हो जाती है तब UV पम्प तथा इन्लेट वॉल्व स्विच ऑफ हो जाते है पम्प निश्चित समय के लिए ड्रेनिंग स्टार्ट कर देता है।

(vi) यदि ड्रेनिंग ओवर हो जाती है तब ड्रेन वॉल्व तथा पम्प स्विच ऑफ हो जाता है तथा स्ट्रीम इ वॉल्व ओपन हो जाता है और सेन्टीजेशन प्रक्रिया प्रारम्भ हो जाती है।

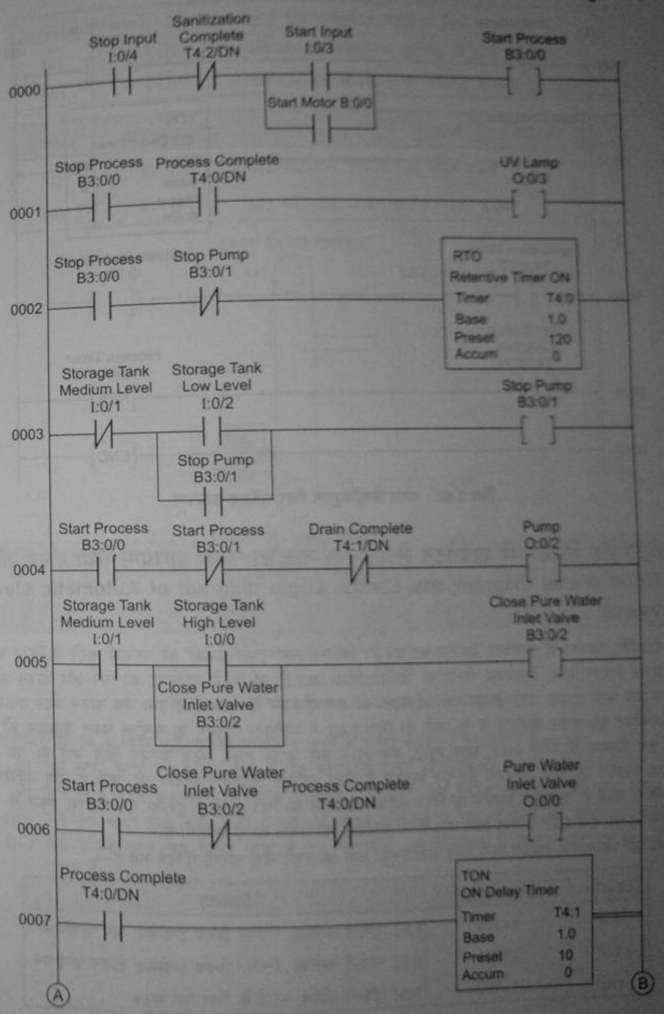
(vii) सेन्टीजेशन के बाद सभी वॉल्व तथा पम्प टर्न ऑफ हो जाते हैं।

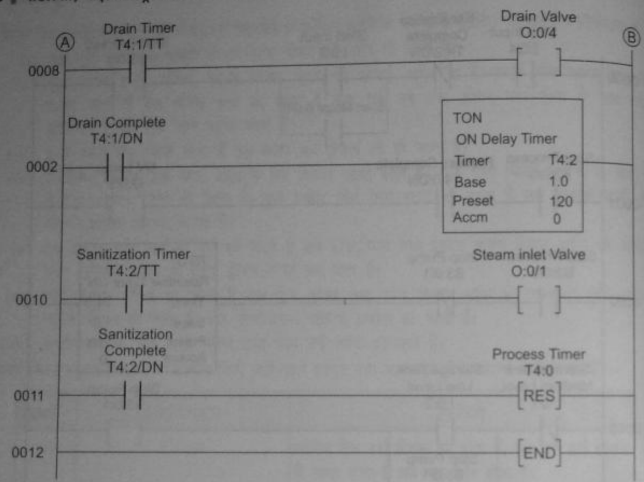
वाटर डिस्ट्रीब्यूशन सिस्टम में उपयोग किये जाने वाले इनपुट तथा आउटपुट नीचे सारणी में दिये गये हैं—

| क्र०सं० | इनपुट एड्रेस | विवरण |
|---|---|---|
| (i) | I : 0/0 | स्टोरेज टैंक हाई लेवल पर होता है। जब यह हाई लेवल से ऊपर होता है तब यह ऑन होता है। |
| (ii) | I : 0/1 | स्टोरेज टैंक मीडियम लेवल पर होता है। यह मीडियम लेवल पर ऑन होता है। |
| (iii) | I : 0/2 | स्टोरेज टैंक लो लेवल पर होता है। यह लो लेवल पर ऑन होता है। |
| (iv) | I : 0/3 | स्टार्ट इनपुट, नॉर्मली ओपन |
| (v) | I : 0/4 | स्टॉप इनपुट, नॉर्मली क्लोज |

| क्र०सं० | आउटपुट एड्रेस | विवरण |
|---|---|---|
| (i) | O : 0/0 | शुद्ध वाटर इन्लेट वॉल्व |
| (ii) | O : 0/1 | स्ट्रीम इन्लेट वॉल्व |
| (iii) | O : 0/2 | पम्प |
| (iv) | O : 0/3 | UV लेम्प |
| (v) | O : 0/4 | ड्रेन वॉल्व |








चित्र 3.62 : वाटर डिस्ट्रीब्यूशन लैडर लॉजिक डायग्राम

## ऑटोमेटिड सिस्टम में एलीवेटर सिस्टम के कन्ट्रोल पेनल डायग्राम तथा लैडर लॉजि डायग्राम (Panel diagram and Ladder Logic diagram of Automatic eleva system)

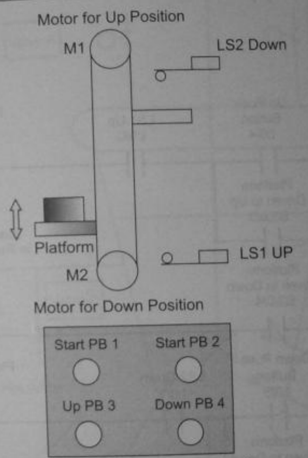
ऑटोमेटिड सिस्टम में एलीवेटर सिस्टम का उपयोग विभिन्न इन्डस्ट्रियल प्रोसेसो को परफॉर्म करने में किया जात इस प्रकार के सिस्टम में दो सिंक्रोनस मोटरों का उपयोग किया जाता है जो किसी ऑब्जेक्ट को अप और डाउन करते इसमें जब अप बटन दाबया जाता है तब यह ऑब्जेक्ट को अप पोजीशन में ले जाती है तथा जब डाउन बटन दबाया है तब ऑब्जेक्ट को डाउन पोजीशन में ले जाती है। चित्र 3.63 में एलीवेटर सिस्टम के कन्ट्रोल पेनल डायग्राम को द गया है। इस सिस्टम में स्टार्ट PB1, स्टार्ट PB2, अप PB3 तथा डाउन PB4 बटन उपयोग किये जाते हैं। यह सि ऑटोमेटिड सिस्टम का एक उदाहरण है। इस सिस्टम में लैडर लॉजिक प्रोग्रामिंग उपयोग की जाती है तथा ऑपरेशन परफॉर्म किया जाता है। एलीवेटर सिस्टम के लैडर लॉजिक डायग्राम को चित्र 3.64 में दर्शाया गया है। इस प्रकार के सि में दो सेन्सर LS1 तथा LS2 उपयोग किये जाते हैं जो ऑब्जेक्ट की स्थिति को सेन्स करते हैं।

एलीवेटर सिस्टम में उपयोग किये आने वाले इनपुट तथा आउटपुट नीचे सारणी में दिये गये हैं—

| क्र०सं० | इनपुट एड्रेस | विवरण |
|---|---|---|
| (i) | I : 0/0 | LS1 नॉर्मली क्लोज्ड, लिमिट स्विच इन्डीकेट अप पॉजीशन |
| (ii) | I : 0/1 | LS2 नॉर्मली क्लोज्ड, लिमिट स्विच इन्डीकेट डाउन पॉजीशन |
| (iii) | I : 0/2 | स्टार्ट नॉर्मली ओपन, स्टार्ट के लिए पुश बटन |
| (iv) | I : 0/3 | स्टार्ट नॉर्मली क्लोज्ड, स्टॉप के लिए पुश बटन |
| (v) | I : 0/4 | अप नॉर्मली ओपन, अप कमाण्ड के लिए पुश बटन |
| (vi) | I : 0/5 | डाउन नॉर्मली ओपन, डाउन कमाण्ड के लिए पुश बटन |



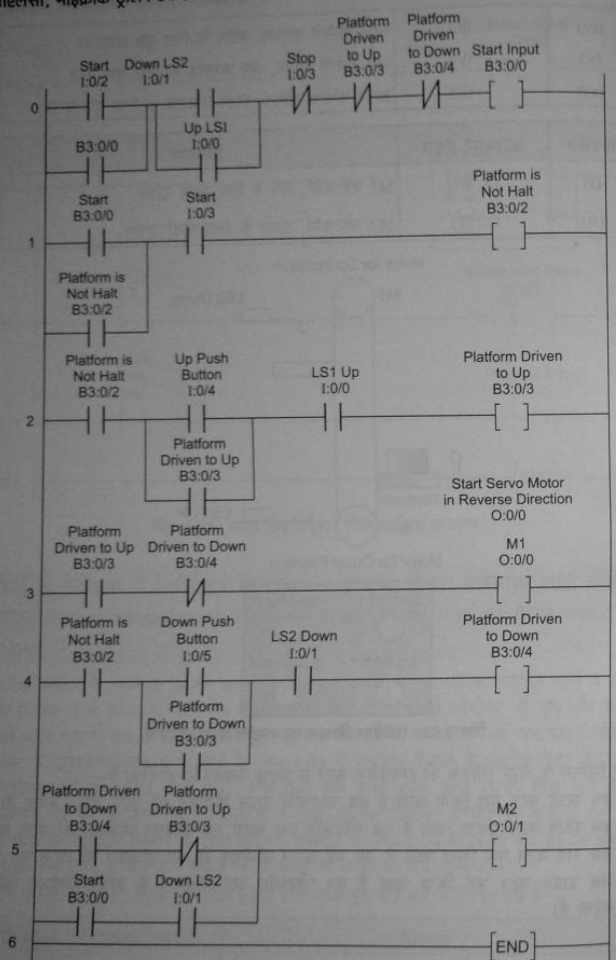
| क्र०सं० | आउटपुट एड्रेस | विवरण |
|---|---|---|
| (i) | O : 0/0 | M1 प्लेटफॉर्म, अप के लिए मोटर ड्राइव |
| (ii) | O : 0/1 | M2 प्लेटफॉर्म, डाउन के लिए मोटर ड्राइव |




चित्र 3.63 : एलीवेटर सिस्टम का कन्ट्रोल पेनल डायग्राम

एलीवेटर सिस्टम के लैडर लॉजिक को इम्प्लीमेन्ट करने के प्रमुख फंक्शन निम्नलिखित हैं—

(i) जब स्टार्ट बटन पुश किया जाता है तब प्लेटफॉर्म डाउन पॉजीशन की तरफ ड्राइव करता है।
(ii) जब स्टॉप बटन दबाया जाता है तब प्लेटफॉर्म उस समय एक निश्चित पॉजीशन को प्राप्त करता है।
(iii) जब अप बटन पुश किया जाता है तब यह डाउन पॉजीशन से अप पॉजीशन की तरफ मूव करता है।
(iv) जब डाउन बटन पुश किया जाता है तब प्लेटफॉर्म अप पॉजीशन से डाउन पॉजीशन की तरफ मूव करता है।

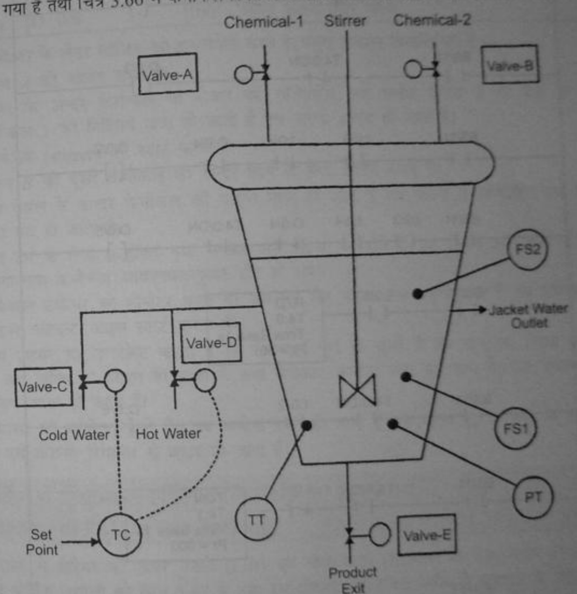



चित्र 3.64 : एलीवेटर सिस्टम का लैडर लॉजिक डायग्राम



## ऑटोमेटिड सिस्टम में केमीकल रियेक्टर का प्रोसेस डायग्राम तथा लैडर लॉजिक डायग्राम (Process diagram and Ladder Logic diagram of Automatic Chemical Reactor)

ऑटोमेटिड सिस्टम के केमीकल रियेक्टर में विभिन्न प्रकार की प्रक्रियाएं सम्पन्न की जाती है। इस प्रकार के ऑपरेशन में सील्ड वेसल में केमीकल को मिक्स किया जाता है तथा इसके टेम्प्रेचर को कोल्ड तथा हॉट वाटर जेकिट की सहायता से कन्ट्रोल किया जाता है, जो वेसल के चारों तरफ होती है। केमीकल रियेक्टर के प्रोसेस डायग्राम को चित्र 3.65 में दर्शाया गया है तथा चित्र 3.66 में केमीकल रियेक्टर के लैडर लॉजिक डायग्राम को दर्शाया गया है।




**Legends :**

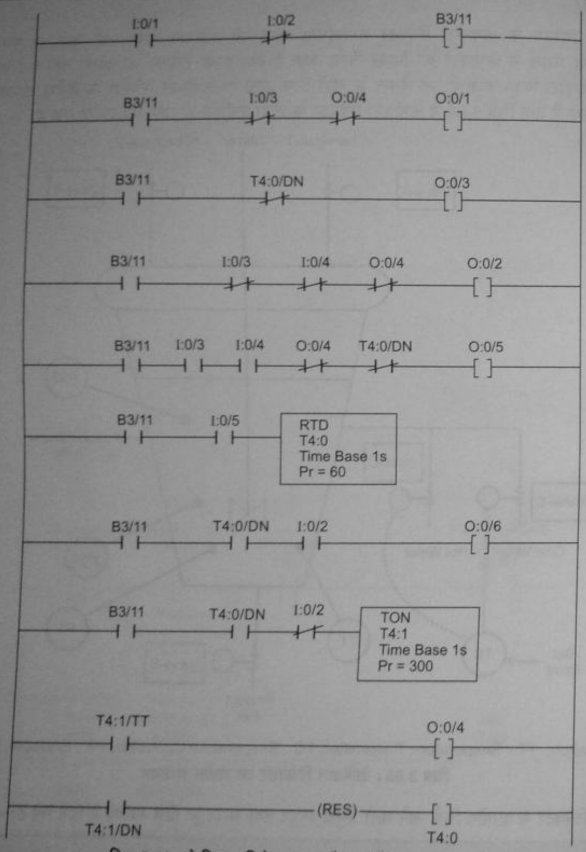
FS : Float Switch; TT : Temperature Transmitter; TC : Temperature Controller; PT : Pressure Transmitter

चित्र 3.65 : केमिकल रियेक्टर का प्रोसेस डायग्राम

केमीकल रियेक्टर में उपयोग किये जाने वाले प्रमुख इनपुट तथा आउटपुट नीचे सारणी में दिये गये है—

| क्र०सं० | इनपुट एड्रेस | विवरण |
|---|---|---|
| (i) | I : 0/1 | टेम्प्रेचर स्टार्ट |
| (ii) | I : 0/2 | सेट पॉइन्ट-1, स्टॉप = I : 0/5 |

| | | |
|---|---|---|
| (iii) | I : 0/3 | टेम्प्रेचर FS1 |
| (iv) | I : 0/4 | सेट पॉइन्ट-2, FS2 = I : 0/6 |



चित्र 3.66 : केमिकल रियेक्टर का लैडर लॉजिक डायग्राम



| क्र०सं० | आउटपुट एड्रेस | विवरण |
|---|---|---|
| (i) | O : 0/1 | वॉल्व A |
| (ii) | O : 0/2 | वॉल्व B |
| (iii) | O : 0/3 | स्टिरर |
| (iv) | O : 0/4 | वॉल्व E |
| (v) | O : 0/5 | टेम्प्रेचर कन्ट्रोलर TC ऑन |
| (vi) | O : 0/6 | 100% ओपन कोल्ड वाटर वॉल्व |

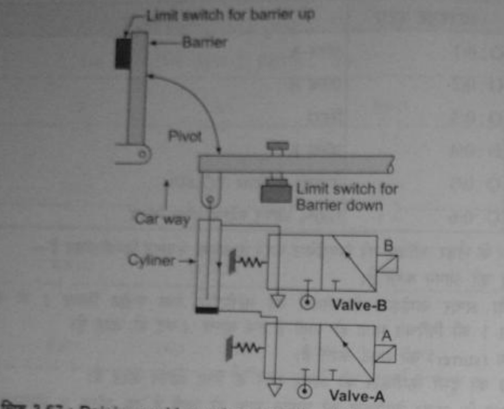
केमीकल रियेक्टर के लैडर लॉजिक को इम्प्लीमेन्ट करने के प्रमुख फंक्शन निम्नलिखित है—

(i) वॉल्व $A$ को ओपन करते हैं।
(ii) वेसल के अन्दर केमीकल के लेवल को मॉनीटरिंग तथा फ्लोट स्विच 1 से चैक करते हैं। जब केमीकल 1 की निश्चित मात्रा हो जाती है तब वाल्व $A$ बंद हो जाता है।
(iii) बिलोडक (stirrer) को स्टार्ट करते हैं।
(iv) वॉल्व $B$ को दूसरे केमीकल को अन्दर लाने के लिए ओपन करते हैं।
(v) जब वेसल के अन्दर केमीकल की पर्याप्त मात्रा हो जाती है तब वॉल्व $B$ मॉनीटरिंग तथा फ्लोट 2 के द्वारा बंद हो जाता है।
(vi) तीन टर्म के लिए कन्ट्रोलर तथा सप्लाई सेट पॉइन्ट पर होते हैं तब स्विच ऑन होता है ताकि मिक्स किया गया केमीकल आवश्यकतानुसार हीट हो सके।
(vii) रियेक्शन टेम्प्रेचर को मॉनीटर करते हैं। जब यह सेट पॉइन्ट तक पहुँच जाता है तब रियेक्शन के दौरान टाइमर, काउन्टर टाइम स्टार्ट करता है।
(viii) जब टाइमर यह इन्डीकेट करता है कि रियेक्शन पूर्ण हो चुकी है तब कन्ट्रोलर स्विच ऑफ हो जाता है तथा वॉल्व $C$ ओपन हो जाता है तथा रियेक्टर कन्टेन्ट ठंडा हो जाता है। इस प्रकार से विलोडक स्विच ऑफ हो जाती है।
(ix) टेम्प्रेचर को मॉनीटर करते हैं। जब कन्टेन्ट ठंडा हो जाता है तब वॉल्व E, 5 मिनट के लिए ऑन होता है एवं प्रोडक्ट रियेक्टर से बाहर हो जाता है।

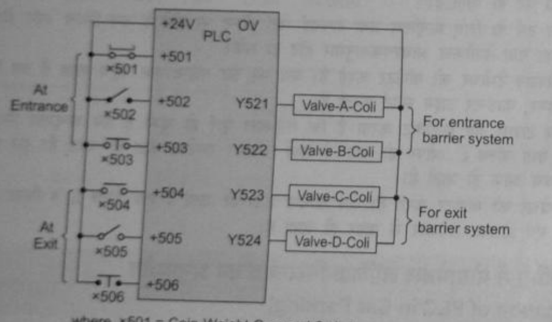
## कार पार्किंग में प्रोग्रामेबल लॉजिक नियन्त्रक का अनुप्रयोग (Application of PLC in Car Parking)

कार के रास्ते में बैरियर को ऊपर उठाने (Lift) एवं नीचे करने (Down) के लिए वाल्व (Valves) प्रणाली प्रयुक्त एक कार पार्किंग प्रणाली को चित्र 3.67 में तथा इस प्रणाली का उचित प्रकार से प्रचालन के लिए प्रयुक्त PLC के संयोजनों को चित्र 3.68 में प्रदर्शित किया गया है। इस प्रणाली में Entrance Barrier तभी खुलता (Open) है जबकि संचयन-बॉक्स (Collection Box) में सही मान युक्त मुद्रा (Correct money) प्रवेश की जाती है तथा Exit-Barrier तभी खुलता है जबकि कार पार्किंग साइड की तरफ, बैरियर से एक निश्चित दूरी पर किसी कार का आवागमन (Detection) होता है।

PLC प्रयुक्त इस कार पार्किंग के Ladder Diagram Programme (LDP) के आरेख को चित्र 3.69 में प्रदर्शित किया गया है। जब संचयन बॉक्स में उचित मान की मुद्रा प्रवेश कराई जाती है तो कम समय के लिए स्विच $X$ 501 ऑन हो जाता है तथा इसकी रिले $Y$ 521 उत्सर्जित (Energized) हो जाती है। इसके फलस्वरूप, वाल्व-$A$ की कुण्डली (Coil) में धारा प्रवाहित होती है; जिससे सिलिण्डर में पिस्टन (Piston), ऊपर की तरफ गति करता है। उसके परिणामस्वरूप, Entrance Barrier अपने Pivot पर घूमता है; जिससे कार के लिए पार्किंग के अन्दर आने का रास्ता




चित्र 3.67 : Raising and Lowering a Barrier of Car Parking System

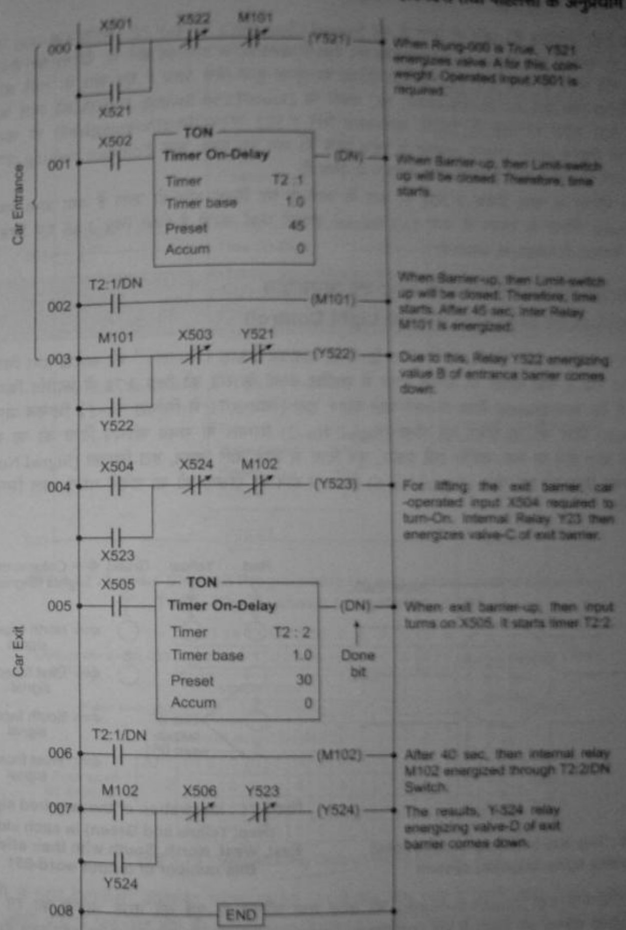



where, ×501 = Coin-Weight Operand Switch
×502 = Limit Switch of the Barrier-Up
×503 = Limit Switch of the Barrier-Down
×504 = Car-Weight Operator Switch
×505 = Limit Switch of the Barrier-Up
×506 = Limit Switch of the Barrier-Down

चित्र 3.68 : PLC Connections of a Car Parking System

स्पर्श करता है तो यह स्विच ऑन हो जाता है; इसके फलस्वरूप टाइमर $T\ 2:1$, समय की गणना करना प्रारम्भ का है तथा 45 sec पश्चात् टाइमर की आउटपुट बिट 0-अवस्था से 1-अवस्था पहुँच जाती है; जोकि आन्तरिक रिले *M* को उत्सर्जित कर देती है। इससे Normal Close (NC) स्विच *M* 101 (जोकि रन्ग-000 में संयोजित है) ऑ जाता है जिससे वाल्व *A* से अनुत्सर्जित (De-energized) हो जाने से Cylinder में Piston अपनी सामान्य अव खुला (Open हो) जाता है। रिले *Y* 521 के ऑन आने पर रन्ग-000 में, स्विच *X* 501 के सामान्तर में सं स्विच-521 के ऑन हो जाने से यह "Latch" का काम करता है। जब बैरियर, Upper Limit-Switch *X* 50






चित्र 3.69 : A Ladder Diagram Programme for Car Parking (Fig. 3.67 and Fig. 3.68)

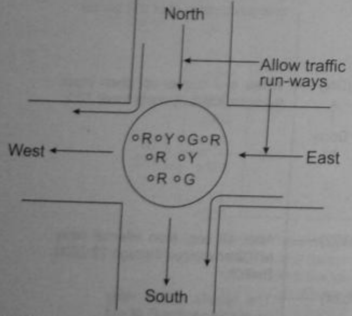
पहुँच जाता है तथा रन्ग-003 में संयोजित स्विच-*M* 101 के ऑन होने से रिले *Y* 522 उत्सर्जित हो जाता है; जिसके फलस्वरूप, सामान्य खुल (Normal Open; NO) स्विच *Y* 522 (जोकि रन्ग-003 में स्विच *M* 101 के समान्तर में

Latch के लिए संयोजित है) ऑन हो जाता है एवं Normal Close (NC) स्विच-Y 22 (जोकि रन्ग-000 मे है) ऑफ हो जाता है तथा Valve-B को उत्सर्जित कर देता है वाल्व B के उत्सर्जित होने से, बैरियर पर नीचे (Down) गति करने के लिए दाब पड़ता है और बैरियर पर घूमता हुआ नीचे आता है एवं कार के रास्ते को देता है। बैरियर जब पूर्ण रूप से नीचे आकर, NC प्रकार के Down-Limit Switch X 503 को स्पर्श करता स्विच X 503 ऑफ हो जाता है; जिसके फलस्वरूप रिले Y 522 अनुत्सर्जित (De-energized) हो जाता वाल्व-B की पिस्टन अपनी सामान्य अवस्था मे पहुँच जाता है। अब यह स्वत: कार Entrance parking प्रणाली कोई कार आने पर अपना कार्य करने के लिए पूर्णरूप से तैयार है।

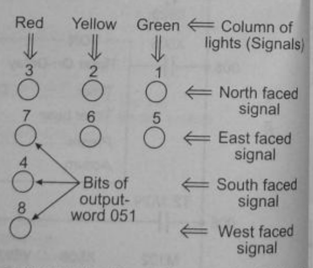
Exit बैरियर के पास, स्विच X 504 पर कार के आने से यह स्विच ऑन हो जाता है तथा अन्य सम्पूर्ण कार Entrance बैरियर के समान ही कार Exit Barrier प्रणाली कार्य करती है जिसे चित्र 3.68 एवं चित्र 3 सहयोग से सरलता से समझा जा सकता है।

## यातायात प्रकाश नियन्त्रण में PLC का अनुप्रयोग (Application of PLC in Traffic Light Control)

उदाहरण के लिए एक "सिक्वेन्सर नियन्त्रित द्वि-रास्ता यातायात प्रकाश (सिग्नल)" को आरेख द्वारा चित्र 3 प्रदर्शित किया गया है तथा चौराहे पर प्रत्येक दिशा में स्थापित बल्बों के रंगों को चित्र 3.71 में प्रदर्शित किया ग इसमें माना है कि उत्तर (North) दिशा से आने वाले वाहन, हरा (चित्र 3.71 में सिग्नल नं०-1) सिग्नल प्राप्त हो दक्षिण (South) दिशा को जा सकेंगे एवं पीला (Signal No.-2) सिग्नल के समय पश्चिम दिशा को जा सकें लाल सिग्नल प्राप्त होने पर रुक जाएँगे। इसी प्रकार, पूर्व दिशा से आने वाले वाहन, हरा सिग्नल (Signal No.-5 होने पर पश्चिम दिशा को एवं पीला (Signal No. 6) के प्राप्त होने पर दक्षिण को जा सकेंगे परन्तु लाल सिग्नल पर रुक जाएँगे।




चित्र 3.70 : Two way (one way at each road) Traffic lights (signals) system

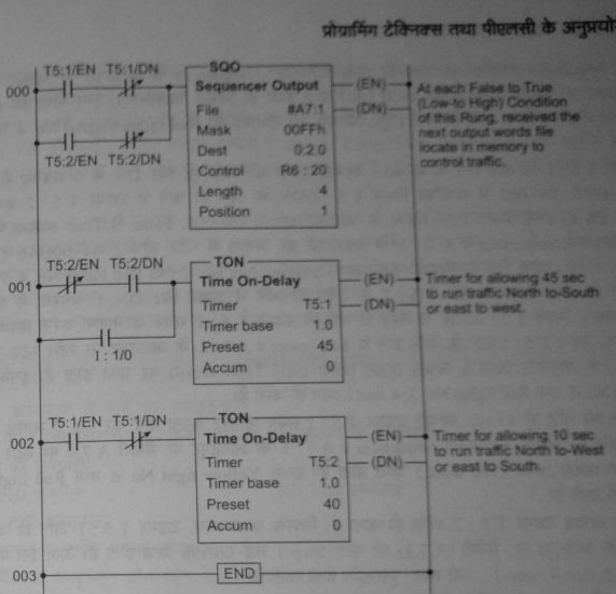



चित्र 3.71 : Illustration of the required signals (Red, Yellow and Green) in each side East, West, North, South with their effective Bits number of output word-051

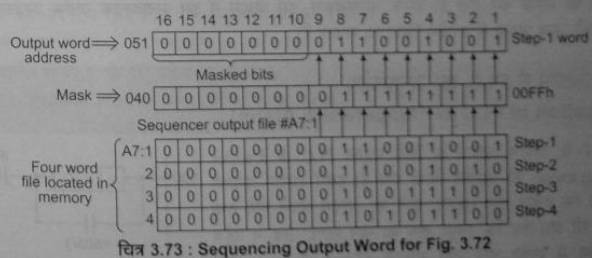
चूँकि दक्षिण-से-उत्तर (South-to-North) की तरफ तथा पश्चिम-से-पूर्व की तरफ वाहनों का रन करने अनुमति है इसलिए पश्चिम एवं दक्षिण में लगे लाल सिग्नल (क्रमश: Signal Nos. 8 and 4) सदैव ऑन रहते हैं।

चित्र 3.70 एवं चित्र 3.71 में प्रदर्शित "Sequence Control Two-way Traffic System" के लैडर डा प्रोग्रामिंग के आरेख को चित्र 3.72 में तथा मैमोरी में 16-बिटों की "Sequencer Output File; # A7 : 1" को 3.73 में प्रदर्शित किया गया है। इसमें Sequencer Output (SQO) निर्देश का उपयोग किया गया है, जोकि प्रकार से प्रणाली को प्रचालित करता है—






चित्र 3.72 : Ladder Diagram Programming for sequencing control of two way Traffic Signals shown in Fig. 3.70 and 3.71




चित्र 3.73 : Sequencing Output Word for Fig. 3.72

मैमोरी मे स्टोर सिक्वेन्सर आउटपुट फाईल (# A7 : 1) में चार स्टेप स्टोर है जिनके द्वारा ट्रैफिक प्रकाश (Traffic Signals), श्रेणी क्रम में प्रकाशित होते हैं। Sequencer Output File को स्टेपिंग प्रदान करने के लिए दो टाइमरों T 5 : 1 एवं T 5 : 2 का प्रयोग किया जोकि क्रमश: 45 sec. एवं 10 sec. पश्चात् Sequencer (SQD) को स्टेपिंग प्रदान करते हैं।

जब रन्ग-001 में संयोजित इनपुट स्विच I : 1 : 0 को ऑन किया जाता है तो टाइमर T 5 : 1, समय की गणना प्रारम्भ कर देता है। इससे T 5 : 1/EN स्विच के ऑन हो जाने से (Rung-000), False से True अवस्था में आ ज

है; जिससे Sequencer प्रचालित हो जाता है। इसके फलस्वरूप, Sequencer के आउटपुट पर Step-1 पर (A7.1), 00C 9h अर्थात् 0000 0000 1100 1001 प्राप्त होता है; जिसके फलस्वरूप, उत्तर दिशा की Green (Signal No.-1) तथा अन्य तीनों दिशाओं (पूर्व, पश्चिम एवं दक्षिण) की Red lights (Signal No. 4, 7 and हो जाती है।

टाइमर T 5 : 1 के ऑन होने के 45 sec. पश्चात् टाइमर की आउटपुट बिट-DN के 0-अवस्था से 1-अ पहुँचने के कारण, रन्ग-002 में संयोजित स्विच T 5 : 1/DN के ऑन हो जाने से टाइमर T 5 : 2 समय को प्रारम्भ कर देता है। इससे स्विच T 5 : 1/EN के ऑन हो जाने से रन्ग-000, False से True अवस्था में आ जिसके फलस्वरूप, sequencer # A 7 : 1 के आउटपुट पर, मैमोरी में स्टोर स्टेप-2 वर्ड-00CAh (अर्थात् 0000 1001 1100) प्राप्त होता है; जिसके फलस्वरूप, Yellow light No.-2 तथा Red lights No. 4, 7 and हो जाती हैं। 10 sec. पश्चात् टाइमर T 5 : 2 के ऑफ हो जाने से इसकी बिट-DN, 1-अवस्था में पहुँच इसके फलस्वरूप, स्विच T 5 : 2/DN के ऑन हो जाने से टाइमर T 5 : 1 समय की गणना करना प्रारम्भ कर जिसके कारण, स्विच T 5 : 1/EN के ऑन होने से Sequencer # A 7 : 1 के आउटपुट पर स्टोर स्टेप-3 वर्ड (चित्र 3.73) के अनुसार, 009Ch अर्थात् 0000 0000 1001 1100 Word) पर प्राप्त होता है। इसके फल Green light No.-5 तथा Red lights No. 3, 4 and 8 ऑन हो जाती है।

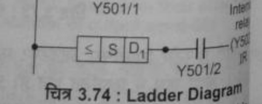
उपरोक्त की भाँति ही 45 sec पश्चात् टाइमर T 5 : 1 समय काउण्ट करना बन्द कर देता है जबकि टाइमर 2 ऑन हो जाता है; जिसके फलस्वरूप Sequencer # A 7 : 1 के आउटपुट पर मैमोरी A 7.4 पर स्टोर वर्ड- (अर्थात् 0000 0000 1010 1100 word) प्राप्त होता है। इससे Yellow light No.-6 तथा Red Lights N and 8 ऑन हो जाती हैं।

10 sec पश्चात् टाइमर T 5 : 2 ऑफ हो जाता है; जिसके फलस्वरूप, टाइमर T 5 : 1 ऑन हो जाता है Sequencer के आउटपुट पर, मैमोरी (A 7.1) पर स्टोर Step-1 वर्ड-00A9h प्राप्त होता है। अतः इस प्रकार step-2, step-3, step-4, step-1... की स्वतः पुनरावृत्ति होती रहती है।

जैसा कि चित्र 3.73 में प्रदर्शित किय गया है कि इसमें मास्क (Mask) मान 00 FFh का उपयोग किया अतः मास्क फाइल उचित डाटा (Light Nos.-1 से 8 तक) के लिये तो Bits को अपने से पास कर देगी जबकि 16 बिट नम्बरों को मास्क कर देती है। इसके फलस्वरूप, इस प्रोग्राम में इस अनावश्यक ब्लॉक आउटपुट डाटा 16 Bits तक) को अन्य प्रोग्राम में उपयोग किया जा सकता है।

## तापीय अलार्म में PLC का अनुप्रयोग (Application of PLC in Temperature Alarm)

चित्र 3.74 में, तापीय अलार्म के लिये प्रयुक्त PLC के Ladder Diagram Programme को प्रदर्शित किया गया है। इसमें अलार्म प्रणाली में हम चाहते हैं कि इनपुट ताप 100°C होने पर अलार्म बजने लगे तथा तब तक बजता रहे जब तक कि इनपुट ताप का मान 80°C तक न पहुँच जाये। इस परिपथ में इनपुट ताप, ताप संवेदक (Temp. Sensor) द्वारा माप कर Source Address (S) पर प्राप्त होता है जबकि अधिकतम ताप सीमा 100°C, प्रथम Destination Address ($D_1$) पर तथा न्यूनतम आवश्यक ताप-सीमा 70°C की द्वितीय Destination Address ($D_2$) पर स्टोर है। इस प्रोग्राम के कारण, यह PLC प्रणाली युक्त तापीय अलार्म निम्न प्रकार से कार्य करेगा—



चित्र 3.74 : Ladder Diagram Programming for PLC used in Temperature Alarm System



चूँकि आन्तरिक रिले (IR) के सम्पर्क (Contacts; IR) सामान्यतः बन्द (Closed) अवस्था में हैं; जब इनपुट ताप (S) का मान, 100°C या इससे अधिक हो जाएगा तो LDP (चित्र 3.74) की रन्ग-000, False से True अवस्था में पहुँचने के कारण अलार्म (Y 501) ऑन हो जाएगा। अलार्म के ऑन होते ही अलार्म से सम्बन्धित सम्पर्क Y 501/1 एवं Y 501/2 ऑन हो जाएँगे। चूँकि अलार्म सम्पर्क Y 501/1, Temp. Data Comparison के एक्रॉस संयोजित हैं अतः Temp का मान 100°C कम होने पर भी रन्ग-000, अपनी सत्य (True) अवस्था में बनी रहेगी, जिससे अलार्म बजता रहेगा।

चूँकि रन्ग-001 से अलार्म सम्पर्क Y 501/2 ऑन अवस्था में है; अतः अब जब इनपुट ताप मान (S) 30°C अथवा इससे कम होने पर आन्तरिक रिले (IR) ऑन हो जाएगा, जिसके कारण रन्ग-001 में संयोजित IR (अर्थात् स्विच IR) ऑफ हो जाएगा; जिसके फलस्वरूप रन्ग-000, True से False अवस्था में पहुँच जाएगी और अलार्म बजना बन्द हो जाएगा। अलार्म Y 501 के बन्द होते ही अलार्म सम्पर्क Y 501/1 एवं Y 501/2 ऑफ हो जायेंगे; इसके फलस्वरूप, परिपथ पुनः अपना कार्य करने के लिए तैयार अवस्था में आ जाएगा।

## अभ्यासीय प्रश्न

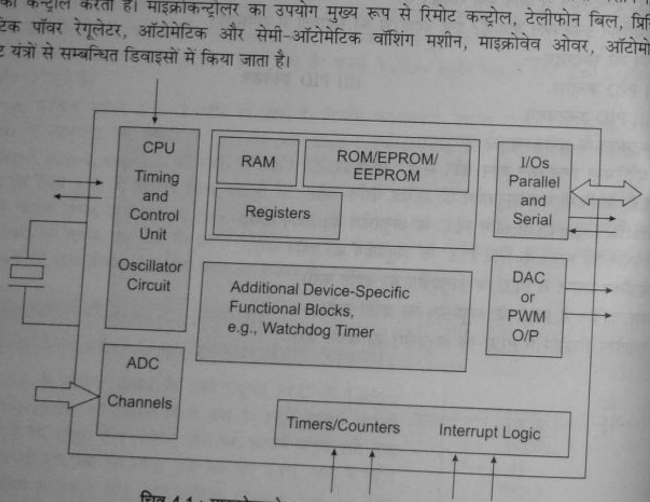
1. PLC इन्स्ट्रक्शन से आप क्या समझते हैं?
2. कम्पेरीजन इन्स्ट्रक्शन क्या हैं? इन्हें उदाहरण की सहायता से समझाइए।
3. निम्न को समझाइए—
   (i) PID कन्ट्रोल (ii) PID इक्वेशन
   (iii) PID इन्स्ट्रक्शन
4. इन्स्ट्रक्शन के वर्गीकरण को समझाइए।
5. मैथमैटिकल इन्स्ट्रक्शन कौन-कौन से हैं? समझाइए।
6. PLC के किसी एक अनुप्रयोग का सचित्र वर्णन करो।
7. Traffic Light Control में PLC के अनुप्रयोग का वर्णन करो।
8. तापमान को मापने के लिए PLC के अनुप्रयोग का वर्णन करो।
9. बोतलिंग प्लान्ट में PLC के अनुप्रयोग का वर्णन करो।
10. कार पार्किंग में PLC के अनुप्रयोग का वर्णन करो।
11. एलीवेटर सिस्टम में PLC के अनुप्रयोग का वर्णन करो।

# 4 माइक्रोकन्ट्रोलर श्रेणी (MCS)-51 का अवलोकन

## [Microcontroller Series (MCS)-51 Overview]

## माइक्रोकन्ट्रोलर (Microcontroller)

माइक्रोकन्ट्रोलर एक सिंगल चिप माइक्रोकम्प्यूटर है, जो विभिन्न प्रकार की ऑटोमेशन तथा मशीन क्रिया प्रोसेस को कन्ट्रोल करता है। माइक्रोकन्ट्रोलर का उपयोग मुख्य रूप से रिमोट कन्ट्रोल, टेलीफोन बिल, प्रिन्टिंग, ऑटोमेटिक पॉवर रेगूलेटर, ऑटोमेटिक और सेमी-ऑटोमेटिक वॉशिंग मशीन, माइक्रोवेव ओवर, ऑटोमोबाइल मेजरमेन्ट यंत्रों से सम्बन्धित डिवाइसों में किया जाता है।




चित्र 4.1 : माइक्रोकन्ट्रोलर का जनरल ब्लॉक डायग्राम

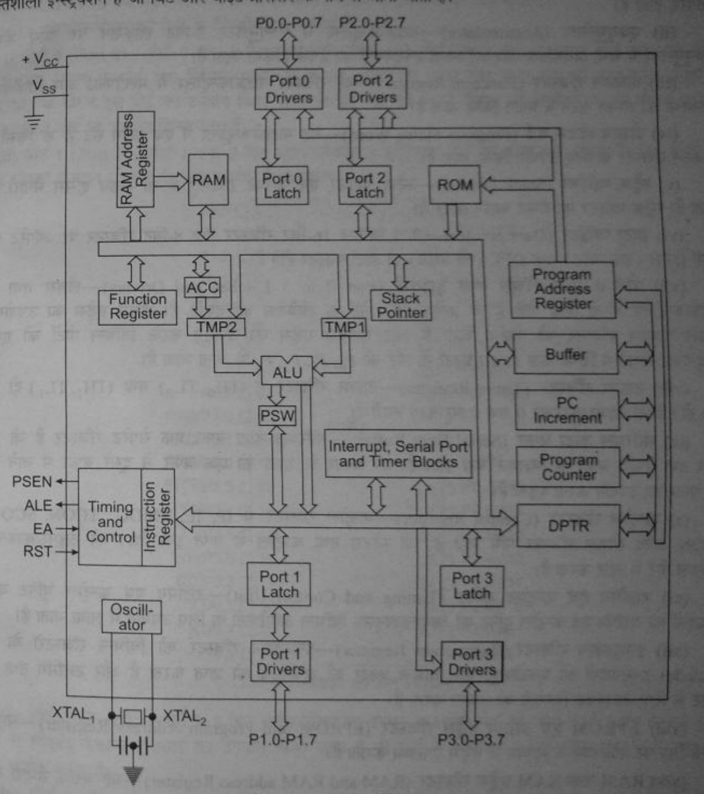
माइक्रोकन्ट्रोलर में सेन्ट्रल प्रोसेसिंग यूनिट, मैमोरी, इनपट/आउटपुट पोर्ट, टाइमर एवं काउन्टर एनालॉग टू डिजीटल कन्वर्टर, डिजीटल टू एनालॉग कन्वर्टर, सीरियल पोर्ट, इन्ट्रप्ट लॉजिक, ऑसीलेटर आदि विभिन्न फंक्शन होते हैं।

इस प्रकार से माइक्रोकन्ट्रोलर में विभिन्न प्रकार के इन्टीग्रेटिड सर्किटों का उपयोग किया जाता है, जो सम्बन्धित विभिन्न क्रियाओं को सम्पन्न करता है। माइक्रोकन्ट्रोलर के उपयोग से किसी भी कार्यक्षेत्र के साइज को किया जा सकता है तथा उसकी कार्यक्षमता को बढ़ाया जाता है। माइक्रोकन्ट्रोलर के उपयोग से ऑटोमेशन की कीमत घटाया जाता है और अधिक से अधिक फ्लेक्सबिलिटी प्रदान की जाती है।



## माइक्रोकन्ट्रोलर का आर्किटेक्चर (Architecture of Microcontroller)

8051 माइक्रोकन्ट्रोलर 8-बिट चिप पर कार्य करता है जो 12 MHz फ्रीक्वेन्सी पर ऑपरेट होता है। यह एक बहुत ही शक्तिशाली इन्स्ट्रक्शन है जो बिट और बाइट प्रोसेसर के नाम से जाना जाता है।




चित्र 4.2 : 8051 आर्किटेक्चर

इस प्रकार 8051 माइक्रोकन्ट्रोलर विभिन्न प्रकार की अर्थमेटिक, बाइनरी, BCD आदि क्रियाओं को प्रोसेस करने की क्षमता रखता है। 8051 माइक्रोकन्ट्रोलर की उत्पत्ति 8031, 8751 आदि माइक्रोकन्ट्रोलर को मिलाने से हुई है। माइक्रोकन्ट्रोलर मुख्यत: तीन पिनों पर कार्य करता है। 8051 माइक्रोकन्ट्रोलर में 128 बाइट डाटा मैमोरी की क्षमता होती है जो विभिन्न रीड/राइट फंक्शनों को प्रचालित करती हैं। माइक्रोकन्ट्रोलर में 20 स्पेशल फंक्शन रजिस्टर्स (SFRs) पाये

जाते हैं। 8051 माइक्रोकन्ट्रोलर के निम्नलिखित ब्लॉक होते हैं—

**(i) ऑसिलेटर** (Osillator)—इस प्रकार के ब्लॉक मे माइक्रोकन्ट्रोलर विभिन्न प्रकार के ऑसिलेटर फंक्शन परफॉर्म करता है। इस फंक्शन मे ऑसिलेटर, क्रिस्टल ऑसिलेटर का उपयोग करता है जो 12 MHz फ्रीक्वेन्सी ऑपरेट होता है।

**(ii) एक्यूमुलेटर** (Accumulator)—माइक्रोकन्ट्रोलर में एक्यूमुलेटर 8-बिट लोकेशन पर कार्य करता एक्यूमुलेटर में सभी अर्थमेटिक और लॉजिकल इन्स्ट्रक्शनों का उपयोग किया जाता है।

**(iii) फंक्शन रजिस्टर** (Function Register)—यह रजिस्टर माइक्रोकन्ट्रोलर में मल्टीप्लाई और डिवीजन क्रियाओं को सम्पन्न करने में प्रयोग किया जाता है।

**(iv) प्रोग्राम स्टेटस वर्ड** (Program Status Word)—यह माइक्रोकन्ट्रोलर में एक फ्लैग सेट है जो किसी फंक्शन रजिस्टर के लिए उपयोग किया जाता है।

**(v) स्ट्रेक प्वॉइंटर** (Stack Pointer)—स्ट्रेक प्वॉइंटर एक 8-बिट रजिस्टर है जो रेन्डम एसिस मैमोरी में जाता है। स्ट्रेक प्वॉइंटर का रीसेट प्वाइंट 07H है।

**(vi) डाटा प्वॉइंटर** (Data pointer)—डाटा प्वॉइंटर 16-बिट रजिस्टर तथा 8-बिट रजिस्टर पर ऑपरेट है इसमें DTH (हाई बाइट) तथा DPL (लो बाइट) दो डाटा प्वाइंटर होते हैं।

**(vii) पोर्ट 0 से 3 लैचस तथा ड्राइवर्स** (Port 0 to 3 Latches and Drivers)—लैचस तथा ड्राइवर्स माइक्रोकन्ट्रोलर में दो जोड़े होते हैं जो प्रत्येक I/O पोर्ट के लोकेशन को दर्शाते हैं। लैचस एड्रेस का उपयोग स्पेशल फंक्शन रजिस्टर को दर्शाया जाता है और ड्राइवर्स एड्रेस का उपयोग करके विभिन्न पोर्टों को एक कम्यूनीकेट कराने में किया जाता है। इस प्रकार के पोर्ट को $P_0, P_1, P_2, P_3$ से जाना जाता है।

**(viii) टाइमर रजिस्टर** (Timer Register)—टाइमर रजिस्टर में $(TH_0, TL_0)$ तथा $(TH_1, TL_1)$ दो पेयर होते हैं जो टाइमर/काउन्टर 0 तथा 1 पर कार्य करते हैं।

**(ix) सीरियल डाटा बफर** (Serial Data Buffer)—सीरियल डाटा बफर एक सेपरेट रजिस्टर है जो बफर तथा रिसीव बफर से मिलकर बना होता है। इस प्रकार के डाटा को एक बफर से दूसरे बफर में जाने के सब बफर का उपयोग करना पड़ता है।

**(x) कन्ट्रोल रजिस्टर** (Control Register)—कन्ट्रोल रजिस्टर में IP, IE, TMOD, TCON, SCON PCON आदि स्पेशल रजिस्टर पाये जाते हैं, जो स्टेटस तथा कन्ट्रोल से प्राप्त इन्फॉर्मेशन को टाइमर/काउन्टर सीरियल पोर्ट में स्टोर करता है।

**(xi) टाइमिंग एवं कन्ट्रोल यूनिट** (Timing and Control Unit)—टाइमिंग एवं कन्ट्रोल यूनिट की डिवाइसों को टाइमिंग एवं कन्ट्रोल यूनिट की आवश्कतानुसार विभिन्न ऑपरेशनों के लिए उपयोग में लाया जाता है।

**(xii) इन्स्ट्रक्शन रजिस्टर** (Instruction Register)—इन्स्ट्रक्शन रजिस्टर को विभिन्न रजिस्टरों के डि (decode) इन्स्ट्रक्शनों को एक्जीक्यूट कर विभिन्न प्रकार की सूचनाओं को प्राप्त करता है और टाइमिंग तथा कन्ट्रोल यूनिट के लिए आवश्यक सिग्नलों को उत्पन्न करता है।

**(xiii) EPROM एवं प्रोग्राम एड्रेस रजिस्टर** (EPROM and Program Address Register)—यह किसी चिप पर मैकेनिज्म के माध्यम से एड्रेस उपलब्ध कराता है।

**(xiv) RAM तथा RAM एड्रेस रजिस्टर** (RAM and RAM address Register)—यह ब्लॉक मैमोरी को बाइट मैकेनिज्म एड्रेस को उपलब्ध कराता है।

**(xv) ए०एल०यू०** (ALU)—यह $TMP_1$ तथा $TMP_2$ रजिस्टर की सहायता से CPU में अर्थमेटिक लॉजिकल ऑपरेशनों को परफॉर्म कराता है।



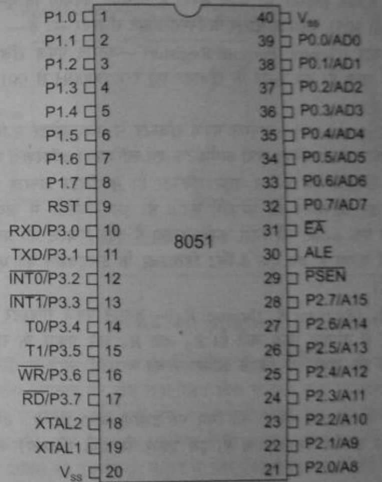
## 8051 माइक्रोकन्ट्रोलर का पिन डायग्राम (Pin Diagram of 8051 Microcontroller)

इंटेल 8051 माइक्रोकन्ट्रोलर में कम्प्यूटर से सम्बन्धित हाई परफर्मिन्स चिप का प्रयोग किया जाता है, जो N-चैनल सिलिकॉन गेट H-MOS तकनीकी पर कार्य करती है। 8051 माइक्रोकन्ट्रोलर में 40 पिनें होती हैं।

8051 माइक्रोकन्ट्रोलर की पिनों के निम्नलिखित फंक्शन हैं—

**(i) पोर्ट 0** (Port 0)—पोर्ट 0 एक 8-बिट बाईडायरेक्शनल I/O पोर्ट है, जो एक आउटपुट पोर्ट की तरह कार्य करता है। इस प्रकार के पोर्ट द्वारा विभिन्न प्रकार के लोअर ऑर्डर के एड्रेस तथा डाटा बस को एक्सटर्नल प्रोग्रामिंग (EPROM) के दौरान इस पोर्ट का उपयोग किया जाता है। इस पोर्ट को किसी प्रोग्राम की वेरीफिकेशन के दौरान ROM तथा EPROM पोर्ट का उपयोग किया जाता है।

**(ii) पोर्ट 1** (Port 1)—पोर्ट 1 एक 8-बिट बाईडायरेक्शनल एड्रेसेबल पोर्ट की तरह कार्य करता है। इस प्रकार पोर्ट को स्पेशल फंक्शन रजिस्टर के लिए उपयोग किया जाता है।




**चित्र 4.3 : 8051 का पिन डायग्राम**

**(iii) पोर्ट 2** (Port 2)—पोर्ट 2, 8-बिट बाईडायरेक्शनल इनपुट/आउटपुट पोर्ट की तरह कार्य करता है। इस प्रकार के पोर्ट में स्पेशल फंक्शन रजिस्टर का उपयोग किया जाता है जो पोर्ट में उपस्थित 8-बिट एड्रेस को ALE तथा पोर्ट पर जनरेट करता है।

**(iv) पोर्ट 3** (Port 3)—यह एक 8-बिट बाईडायरेक्शनल I/O पोर्ट है जो माइक्रोकन्ट्रोलर की आन्तरिक क्रियाओं को सम्पन्न करता है। इसमें आउटपुट बफर सिंक होता है जो इनपुट LSTTL पर कार्य करता है।

**(v) ALE/PROG**—यह एक एड्रेस लैच इनेबल के द्वारा परिभाषित पोर्ट है। यह एक्स्टर्नल मैमोरी में लो एड्रेस लॉचिंग के लिए उपयोग किया जाता है। इस प्रकार के पोर्ट में ALE आठ LSTTL इनपुट रख सकता है।

**(vi) EA/$V_{PP}$**—यह एक्सटर्नल एसिस इनेबल पिन है जो 8051 माइक्रोकन्ट्रोलर की मैमोरी में विभिन्न को रखती है। इसमें $\overline{EA}$ लो पॉवर पर विभिन्न प्रकार के प्रोग्रामों को एक्सटर्नल मैमोरी में एक्जीक्यूट करता है।

प्रोग्राम एक्सटर्नल मैमोरी में एक्जीक्यूट होता है तब $\overline{EA}$ हाई हो जाता है। यह 21 वोल्ट पर EPROM चिप पर रिसीव करता है।

(vii) $\overline{PSEN}$—इस पिन को प्रोग्राम स्टोरेबल इनेबल कहा जाता है जो लो आउटपुट सिग्नल पर एक्स मैमोरी में विभिन्न प्रोग्रामों को स्टोर करता है।

इस प्रकार यह कहा जा सकता है कि 8051 माइक्रोकन्ट्रोलर में 40 पिनें होती हैं जो अलग-अलग फंक्शन परफॉर्म करती हैं। सभी पिनें एक चिप के रूप में पायी जाती हैं, जो माइक्रोकन्ट्रोलर की एक्सटर्नल मैमोरी में वि प्रकार के प्रोग्रामों को एक्जीक्यूट करती हैं। सभी पिनें अपने अलग-अलग कार्यों को विभिन्न इनपुट/आउटपुट फं की सहायता से प्रोग्रामों को प्रदर्शित करती हैं।

## 8051 माइक्रोकन्ट्रोलर के रजिस्टर सेट (Register set of 8051 Microcontroller)

8051 माइक्रोकन्ट्रोलर 8-बिट रजिस्टर पर कार्य करता है जिसका उपयोग विभिन्न प्रयोग के इन्स्ट्रक्शन सेट ऑपरेट कराने में किया जाता है। 8051 माइक्रोकन्ट्रोलर के निम्नलिखित रजिस्टर सेट हैं—

**(i) जनरल पर्पज रजिस्टर** (General Purpose Register)—जनरल पर्पज रजिस्टर का उपयोग 2 बाइट एक्सेस मैमोरी के लिए किया जाता है। इस प्रकार के रजिस्टर की रेन्ज 0000H से 001FH तक होती है। जनरल रजिस्टर के निम्नलिखित बिन्दु हैं—

**( अ ) एक्यूमुलेटर** (Accumulator)—जनरल पर्पज रजिस्टर में एक्यूमुलेटर 8-बिट रजिस्टर पर कार्य कर एक्यूमुलेटर, जनरल पर्पज रजिस्टर से सम्बन्धित सभी अर्थमेटिक एवं लॉजिकल ऑपरेशनों को परफॉर्म करता है।

**( ब ) *B*-रजिस्टर**—*B*-रजिस्टर एक 8-बिट बाइट रजिस्टर है। *B*-रजिटर जनरल पर्पज रजिस्टर में पाया जा जो मल्टीप्लीकेशन और डिवीजन ऑपरेशन को परफॉर्म करता है। इस रजिस्टर में जब मल्टीप्लीकेशन इन्स्ट्रक्श उपयोग किया जाता है तब यह एक 8-बिट को स्टोर करके रखता है और रिजल्ट को प्रदर्शित करता है। इसी प्रका डिवीजन ऑपरेशन को परफॉर्म करता है तब एक 8-बिट डिवाइजर के इन्स्ट्रक्शन को एक्जीक्यूट करता है और को 8-रजिस्टर में स्टोर करता है।

**( स ) रजिस्टर $R_0$ से $R_7$** (Register $R_0$ through $R_7$)—जनरल पर्पज रजिस्टर में $R_0$ से $R_7$ तक 8 र होते हैं जो स्क्रेच पैड रजिस्टर में उपयोग किये जाते हैं। $R_0$ तथा $R_7$ चार प्रकार के रजिस्टरों की एक बैंक है 8-बिट बाइट रजिस्टर होते हैं। इस प्रकार से किसी ऑपरेशन को परफॉर्म करने के लिए प्रोग्राम स्टेटस वर्ड (PS उपयोग किया जाता है।

इस प्रकार के रजिस्टरों को RAM मैमोरी की चिप पर दर्शाया जाता है। $R_0$ तथा $R_7$ रजिस्टरों में प्रोग्राम लिखते समय 32 रजिस्टरों का उपयोग किया जाता है। इस प्रकार से सभी रजिस्टरों के इन्स्ट्रक्शन सेट को मैमो प्रदर्शित किया जाता है।

**(ii) स्टेक प्वाइंटर तथा प्रोग्राम काउन्टर** (Stack Pointer and Program Counter)—स्टेक प्वॉइंटर 8 माइक्रोकन्ट्रोलर में एक 8-बिट बाइट रजिस्टर होता है। स्टेक प्वाइंटर रजिस्टर में डाटा push और call ऑपरेशन बढ़ावा दिया जाता है तथा pop और return ऑपरेशन को घटाया जाता है। स्टेक प्वॉइंटर RESET ऑपरेशन को 0 तथा 0HH पर प्रदर्शित करता है।

प्रोग्राम काउन्टर ऑपकोड बाइट इन्स्ट्रक्शन का उपयोग करके विभिन्न प्रकार के एड्रसों को मैमोरी पर प्रद करता है। 8051 माइक्रोकन्ट्रोलर में प्रोग्राम काउन्टर 16-बिट बाइट का होता है। प्रोग्राम काउन्टर में call और ju इन्स्ट्रक्शनों का उपयोग किया जाता है। प्रोग्राम काउन्टर के रजिस्टर एड्रेस RAM मैमोरी चिप पर उपलब्ध होते हैं।

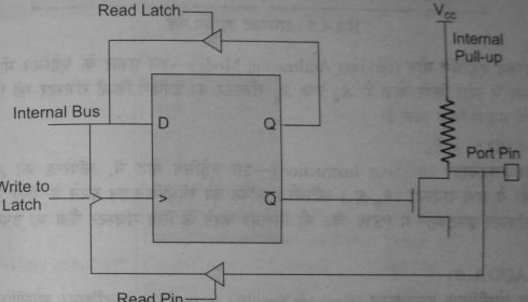
**(iii) स्पेशल फंक्शन रजिस्टर** (SFRs)—स्पेशल फंक्शन रजिस्टर 128 बाइट मैमोरी लोकेशन को 08H OFFH लोकेशन पर प्राप्त किया जाता है। इस प्रकार 08H से OFFH लोकेशन की सहायता से विभिन्न स्पेशल फं प्राप्त किये जाते हैं जिन्हें स्पेशल फंक्शन रजिस्टर कहते हैं। इन सभी प्रकार के स्पेशल फंक्शन का उपयोग 80



माइक्रोकन्ट्रोलर के फंक्शन को कन्ट्रोल तथा उसकी स्थिति का पता लगाने में किया जाता है। माइक्रोकन्ट्रोलर के सभी स्पेशल फंक्शन रजिस्टरों को आसानी से पढ़ा व लिखा जा सकता है। स्पेशल फंक्शन रजिस्टर में PSW, TMOD, TCON, SCON, PCON आदि रजिस्टर होते हैं जो विभिन्न फंक्शनों में उपयोग किये जाते हैं।

## माइक्रोकन्ट्रोलर के इनपुट/आउटपुट (I/O) पोर्ट स्ट्रक्चर (Input/Output (I/O) Port Structure of Microcontroller)

माइक्रोकन्ट्रोलर के सभी फंक्शनों में इनपुट/आउटपुट पोर्ट का उपयोग किया जाता है। I/O पोर्ट, 8051 माइक्रोकन्ट्रोलर के 8-बिट पोर्ट पर कार्य करता है। I/O पोर्ट एक एड्रेसेबल पिन की तरह उपयोग में लाया जाता है। प्रत्येक I/O पोर्ट में लैच फंक्शन का उपयोग किया जाता है जो आउटपुट ड्राइवर तथा इनपुट बफर के द्वारा ऑपरेट होता है।




चित्र 4.4 : I/O पोर्ट स्ट्रेक्चर

माइक्रोकन्ट्रोलर के इस I/O पोर्ट में *D*-फ्लिप-फ्लॉप का उपयोग किया जाता है जो CPU से सम्बन्धित विभिन्न आन्तरिक बसों के माध्यम से लैच सिग्नल को रीड तथा राइट लैच फंक्शन तक पहुँचाता है। I/O पोर्ट में *Q* आउटपुट को फ्लिप-फ्लॉप के माध्यम से इन्टरनल डाटा बस तक पहुँचाता है। रीड पिन का उपयोग लैच फंक्शनों के विभिन्न ऑपरेशनों को पढ़ने में किया जाता है। पोर्ट पिन एक कमाण्ड है जो इन्टरनल डाटा बस के माध्यम से CPU में दी जाती है। माइक्रोकन्ट्रोलर के किसी भी प्रोग्राम को एक्जीक्यूट कराने के लिए निम्नलिखित I/O पोर्ट उपयोग किये जाते हैं—

**(i) पोर्ट 1 तथा 3**—पोर्ट 1, 2 तथा 3 को बाइडायरेक्शनल पोर्ट कहा जाता है, जो विभिन्न प्रकार के pull-up रजिस्टरों का उपयोग करता है। इस प्रकार के पोर्टों का उपयोग इनपुट पिन की तरह किया जाता है। माइक्रोकन्ट्रोलर में प्रोग्राम को लिखते समय इनपुट पोर्ट को '1' तथा आउटपुट पोर्ट को '0' से दर्शाया जाता है।

**(ii) पोर्ट 0**—पोर्ट 0 को आउटपुट पोर्ट कहा जाता है जो FET ऑपरेशन पर कार्य करता है। पोर्ट 0 में इन्टरनल pull-up रजिस्टर नहीं पाया जाता है। इसमें जब किसी प्रोग्राम को एक्जीक्यूट किया जाता है तब पोर्ट 0 को '1' बैच सिग्नल से दर्शाया जाता है। यह सिग्नल जब FET पर जाता है तब FET, OFF हो जाता है। इस प्रकार के पोर्ट में एक्सटर्नल मैमोरी का उपयोग किया जाता है जो 8LS TTL इनपुट के रूप में ऑपरेट होता है।

इस प्रकार के I/O पोर्ट को माइक्रोकन्ट्रोलर का मुख्य ऑपरेशन कहा जाता है जो सभी प्रकार के माइक्रोकन्ट्रोलर से सम्बन्धित प्रोग्रामों में उपयोग किया जाता है।

## माइक्रोकन्ट्रोलर के एड्रेसिंग मोड्स (Addressing Modes of Microcontroller)

**(i) डायरेक्ट एड्रेसिंग मोड** (Direct Addressing Mode)—डायरेक्ट एड्रेसिंग मोड स्पेशल इंस्ट्रक्शन ऑपरेट करता है। इस प्रकार के एड्रेसिंग मोड में 128 बाइट इन्टरनल रेम एवं स्पेशल फंक्शन रजिस्टर का उप किया जाता है।

**उदाहरण**—MOV A, Direct इन्स्ट्रक्शन का उपयोग सोर्स ऑपरेन्ट की तरह किया जाता है। MOV A, 54H उपयोग चिप मैमोरी लोकेशन के लिए किया जाता है। MOV A, 5BUF का उपयोग एक्यूमुलेटर में स्पेशल फंक रजिस्टर के लिए किया जाता है।

| | |
|---|---|
| Opcode Byte | 8-bit Direct Address |

चित्र 4.5 : डायरेक्ट एड्रेसिंग मोड

**(ii) इनडायरेक्ट एड्रेसिंग मोड** (Indirect Addressing Mode)—इस प्रकार के एड्रेसिंग मोड में 8-बिट ए ऑपरेन्ड को रजिस्टर में स्टोर किया जाता है। $R_0$ तथा $R_1$ रजिस्टर का उपयोग किसी रजिस्टर की स्थिति ज्ञात करने लिए स्टैक प्वाइंट के रूप में किया जाता है।

**उदाहरण**—ADD A, @ $R_0$

**(iii) रजिस्टर इन्स्ट्रक्शन** (Register Instruction)—इस एड्रेसिंग मोड में, ऑपरेन्ड को $R_0$-$R_7$ रजिस्टर स्टोर किया जाता है। ये सभी रजिस्टर ($R_0$-$R_7$) ऑपकोड फॉर्मेट को स्पेसीफिकेशन करने के लिए 3-बिट रजिस्टर उपयोग करते हैं। रजिस्टर इन्स्ट्रक्शन में PSW बिट को सिलेक्ट करने के लिए रजिस्टर बैंक का उपयोग किया जाता

**उदाहरण**—ADD A, $R_7$

**(iv) रजिस्टर स्पेसीफिक इन्स्ट्रक्शन** (Register Specific Instruction)—रजिस्टर स्पेसीफिक इन्स्ट्रक्शन को ऑपरेट करने के लिए स्पेसीफाइड रजिस्टर का उपयोग किया जाता है। इस प्रकार स्पेसीफाइड रजिस्टर को ऑप करने के लिए कुछ इन्स्ट्रक्शनों का उपयोग किया जाता है। इन सभी प्रकार के इन्स्ट्रक्शनों को एक श्रेणी में रखा ज है।

**उदाहरण**—*RLA*, इन्स्ट्रक्शन का उपयोग एक्यूमुलेटर के बाँयी साइड किया जाता है।

**(v) इमीडियेट एड्रेसिंग मोड** (Immediate Addressing Mode)—इस प्रकार के एड्रेसिंग मोड का उप इमीडियेट डाटा के लिए किया जाता है। इसमें विभिन्न फंक्शनों के लिए स्पेसीफाइड इन्स्ट्रक्शन उपयोग में लाये जाते हैं।

**उदाहरण**—MOV A, # 100.

**(vi) इन्डेक्सड एड्रेसिंग मोड** (Indexed Addressing Mode)—इन्डेक्सड एड्रेसिंग मोड में प्रोग्राम काउन्टर अ डाटा प्वाइंटर को 16-बिट एड्रेस रजिस्टर को स्टोर करने में उपयोग किया जाता है। इस प्रकार के मोड में 16-बि रजिस्टर को एक्यूमुलेटर टेबिल की सहायता से देखा जा सकता है। इन्डेक्सड एड्रेसिंग मोड एक्यूमुलेटर का एक फंक्श है जो 16-बिट रजिस्टर के JUMP डेस्टीनेशन एड्रेस में स्टोर किया जाता है।

**उदाहरण**— MOV CA, @ A + DPTR.

JMP @ A + DPTR



## स्पेशल फंक्शन रजिस्टर तथा रेन्डम एक्सेस मैमोरी (Special Function Register and Random Access Memory)

### स्पेशल फंक्शन रजिस्टर (Special Function Register)

स्पेशल फंक्शन रजिस्टर में मैमोरी लोकेशन 08H से OFF को 128 बाइट मैमोरी लोकेशन के लिए विभिन्न फंक्शनों का उपयोग किया जाता है, जिन्हें स्पेशल फंक्शन रजिस्टर कहते हैं। इन सभी प्रकार के स्पेशल फंक्शन रजिस्टरों का उपयोग माइक्रोकन्ट्रोलर के फंक्शनों को कन्ट्रोल करने तथा उनकी स्थिति का पता लगाने के लिए किया जाता है। माइक्रोकन्ट्रोलर के स्पेशल फंक्शन रजिस्टर मे PSW, TCON, TMOD, SCON, PCON आदि रजिस्टर होते है, जिन्हें प्रोग्राम में उपयोग करने के लिए आसानी से पढ़ा व लिखा जा सकता है। इस प्रकार से स्पेशल फंक्शन रजिस्टरों को निम्नलिखित सारणी में उनके नाम, प्रतीक, एड्रेस को दर्शाया गया है—

| SFR symbol | Register name | Address |
|---|---|---|
| ACC* | Accumulator | 0E0H |
| B* | B-register | 0F0H |
| P0* | Port 0 | 80H |
| P1* | Port 1 | 90H |
| P2* | Port 2 | 0A0H |
| P3* | Port 3 | 0B0H |
| IP* | Interrupt Priority Control | 0B8H |
| IE* | Interrupt Enable Control | 0A8H |
| TMOD | Timer/Counter Mode Control | 89H |
| T2CON (Only in 8052)* | Timer/Counter 2 Control | 0C8H |
| TCON* | Timer/Counter Control | 88H |
| TH0 | Timer/Counter 0 (high byte) | 8CH |
| TL0 | Timer/Counter 0 (low byte) | 8AH |
| TH1 | Timer/Counter 1 (high byte) | 8DH |
| TL1 | Timer/Counter 1 (low byte) | 8BH |
| TH2 (Only in 8052) | Timer/Counter 2 (high byte) | 0CDH |
| TL2 (Only in 8052) | Timer/Counter 2 (low byte) | 0CCH |
| RCAP2H (Only in 8052) | Timer/Counter 2 Capture Register (high byte) | 0CBH |
| RCAP2L (Only in 8052) | Timer/Counter 2 Caputre Register (low byte) | 0CAH |
| SCON* | Serial Control | 98H |
| SBUF | Serial Data Buffer | 99H |
| PCON | Power Control | 97H |
| PSW* | Program Status Word | 0D0H |
| DPTR | Data Pointer | |
| DPH | Data Pointer (high byte) | 83H |
| DPL | Data Pointer (low byte) | 82H |

स्पेशल फंक्शन रजिस्टर में निम्नलिखित रजिस्टर हैं—

**(i) प्रोग्राम स्टेटस वर्ड** (Program Status Word)—प्रोग्राम स्टेटस वर्ड (PSW) एक साधारण 8-बिट रजि[स्टर है] जो कैरी, ऑक्सिलियरी कैरी, ऑवरफ्लो और पेरटी फ्लैग आदि के द्वारा विभिन्न प्रकार के फंक्शनों को परफॉर्म क[रता] है। बिट RS 1 तथा RS 0 रजिस्टर बैंक सिलेक्शन में उपयेाग की जाती हैं। प्रोग्राम स्टेटस वर्ड एक एड्रेसेबल रजि[स्टर] है। इस प्रकार के प्रोग्राम स्टेटस वर्ड में PSW.0 लीस्ट सिग्नीफिकेन्ट बिट (LSB) का उपयोग पेरटी फ्लैग में कर[ता] और मोस्ट सिग्नीफिकेन्ट बिट (MSB) का उपयोग कैरी फ्लैग में करता है।

| MSB | | | | | | | LSB |
|---|---|---|---|---|---|---|---|
| C | AC | F0 | RS1 | RS0 | OV | – | P |

**चित्र 4.6 : प्रोग्राम स्टेटस वर्ड**

प्रोग्राम स्टेटस वर्ड के निम्नलिखित भाग होते हैं—

**(अ) कैरी फ्लैग** (Carry Flag)—कैरी फ्लैग अर्थमैटिक एवं लॉजिकल ऑपरेशन को परफॉर्म करता है, उससे प्राप्त रिजल्ड को PSW 7 को 7th बिट में दर्शाता है।

**उदाहरण**—दो 8-बिट कैरी फ्लैग नम्बरों को घटाने व जोड़ने से जो कैरी प्राप्त होती है, उसे कैरी फ्लैग में द[र्शाया] जाता है। इस प्रकार OC 2H (1100001B) तथा OAAH (101010B) को जोड़ने पर प्राप्त कैरी को निम्नलि[खित] प्रकार से दर्शाया जाता है—

```
              11000010B
            + 10101010B
Carry → [1]   01101100B
```

इस प्रकार कैरी को [1] से दर्शाया गया है।

**(ब) FO** – FO जनरल पर्पज रजिस्टर का एक यूजर है जो इस रजिस्टर के सॉफ्टवेयर में सेट रहता है। इ[स] स्थिति का पता सॉफ्टवेयर से ही लगाया जाता है।

**(स) रजिस्टर बैंक सिलेक्ट बिट RS1 एवं RS0** (Register Bank Select Bits RS 1 and RS 0)—[इस] प्रकार के रजिस्टर बैंक में चार रजिस्टर बैंक सिलेक्ट की जाती है। प्रत्येक रजिस्टर बैंक R0 से R7 तक होत[ी है।] रजिस्टर बैंक RS0 तथा RS1 की एड्रेस रेन्ज तथा रजिस्टर बैंक को निम्नलिखित सारणी में दर्शाया गया है—

| RS1 | RS0 | Register Bank Selected | Address Rang into the Ch[ip] |
|---|---|---|---|
| 0 | 0 | Bank 0 | 00 - 07H |
| 0 | 1 | Bank 1 | 08 - OFH |
| 1 | 0 | Bank 2 | 10 - MH |
| 1 | 1 | Bank 3 | 18 - 1FH |

**(द) ओवरफ्लो फ्लैग** (Overflow Flag)—ओवरफ्लो फ्लैग का उपयोग अर्थमेटिक ऑपरेशन (जोड़, घटा[व,] गुणा और भाग) में किया जाता है। इस प्रकार के फ्लैग को कैरी 6 बिट तथा 7 बिट के द्वारा प्रदर्शित किया जाता [है।] इस प्रकार 1000010 B तथा 10101010 B को जोड़ने पर प्राप्त कैरी 7 बिट में जायेगी 6 बिट में नहीं जाये[गी] इसलिए ओवरफ्लो फ्लैग को कैरी फ्लैग के साथ जोड़ा जाता है।

**(ii) डाटा प्वॉइंटर** (Data Pointer)—डॉटा प्वॉइंटर (DDTR) एक 16-बिट रजिस्टर है जो दो बाइट्स [से] मिलकर बना है। इस प्रकार के डाटा प्वॉइंटर में उच्च बाइट को DPH तथा निम्न बाइट को DPL से प्रदर्शित किया [जाता है।]



डाटा प्वॉइंटर को एड्रेसिंग ऑफ-चिप के डाटा तथा कोड में MOV X एवं MOV C कमाण्ड का उपयोग किया जाता है। इस प्रकार से डाटा प्वॉइंटर में 64 k ऑफ-चिप डाटा मैमोरी और 64 k ऑफ-चिप प्रोग्राम मैमोरी का उपयोग मैक्सिमम (maximum) एड्रेसेबल के लिए किया जाता है। डाटा प्वाइंटर में "INC DPTR" इंस्ट्रक्शन का उपयोग कर 16-बिट DTPR की मात्रा को बढ़ाया जाता है। इस प्रकार के प्वॉइंटर में जनरल पर्पज रजिस्टर का उपयोग किया जाता है।

**(iii) टाइमर रजिस्टर** (Timer Register)—टाइमर रजिस्टर के रजिस्टर पेयर (TH0, TL0), (TH1, TL1), (TH2, TL2) को 16-बिट के टाइमर/काउन्टर रजिस्टर 0, 1, 2 से प्रदर्शित कियाजाता है। इस प्रकार के इन्स्ट्रक्शनों का उपयोग रजिस्टरों की बिट-साइज रीडिंग तथ राइटिंग में किया जाता है। इस ऑपरेशन का प्रयोग केवल टाइमिंग और काउन्टिंग के लिए किया जाता है। टाइमिंग रजिस्टर में विभिन्न मोड पाये जाते हैं जो टाइमर का विवरण देते हैं। टाइमर रजिस्टर में टाइमर के वितरण के लिए TCON और TMOD टाइमर मोड का उपयोग किया जाता है।

**(iv) पोर्ट 0 से 3** (Port 0 to 3)—स्पेशल फंक्शन रजिस्टर में P0, P1, P2, P3 चार प्रकार के I/O पोर्ट पाये जाते हैं। इसमें प्रत्येक पोर्ट बिट एड्रेसेबल तथा बाइट एड्रेसेबल है।

**(v) कन्ट्रोल रजिस्टर** (Control Register)—स्पेशल फंक्शन रजिस्टर में TCON, TMOD, IE, IP, SCON, PCON को कन्ट्रोल एवं स्टेटस इन्ट्रप्टस, सीरियल I/O और टाइमर/काउन्टर के रूप में उपयोग किया जाता है।

**(vi) कैप्त्रर रजिस्टर** (Capture Register)—कैप्चर रजिस्टर, रजिस्टर RCAP2H तथा RCAP2L के दो पेयर रजिस्टर हैं। कैप्चर मोड ऑपरेशन को 8052 माइक्रोकन्ट्रोलर की सहायता से प्रदर्शित किया जाता है। इसमें दो या दो से अधिक रजिस्टरों का ट्रांजेक्शन 8052 T2EX की पिनों $TH_2$ और $TL_2$ के द्वारा होता है।

## रेन्डम एक्सेस मैमोरी (RAM)

माइक्रोकन्ट्रोलर की मैमोरी कम्प्यूटर पर आधारित होती है। कम्प्यूटर में उच्चगति की मुख्य मैमोरी निश्चित लम्बाई के कम्प्यूटर शब्दों में विभक्त होती है। मैमोरी $N$ शब्दों में विभक्त होती है तथा प्रत्येक शब्द $M$ बिट्स में स्टोर किये जाते हैं। इस प्रकार से मैमोरी के प्रत्येक शब्द को एक एड्रेस दिया जाता है, इस एड्रेस पर उस शब्द को प्राप्त किया जाता है।

इस प्रकार मुख्य मैमोरी में एड्रेस 0, एड्रेस 1, एड्रेस 2 क्रमश: मैमोरी के शब्द हैं तथा प्रत्येक एड्रेस पर शब्द स्टोर किया जाता है।

माइक्रोकन्ट्रोलर की मैमोरी में स्टोर, जिस डाटा को प्राप्त करना होता है अथवा उसे लिखना होता है। उस लोकेशन का एड्रेस, मैमोरी एड्रेस रजिस्टर में स्टोर किया जाता है। मैमोरी एड्रेस रजिस्टर मे $n$ बिट्स स्टोर किये जा सकते हैं। माइक्रोकन्ट्रोलर की मैमारी में जिस डाटा को मैमारी लोकेशन में स्टोर करना होता है उसे मैमोरी बफर रजिस्टर में रखा जाता है। मैमोरी बफर रजिस्टर में $m$ बिट्स स्टोर किये जा सकते हैं। माइक्रोकन्ट्रोलर की RAM मैमोरी के निम्नलिखित बिन्दु हैं—

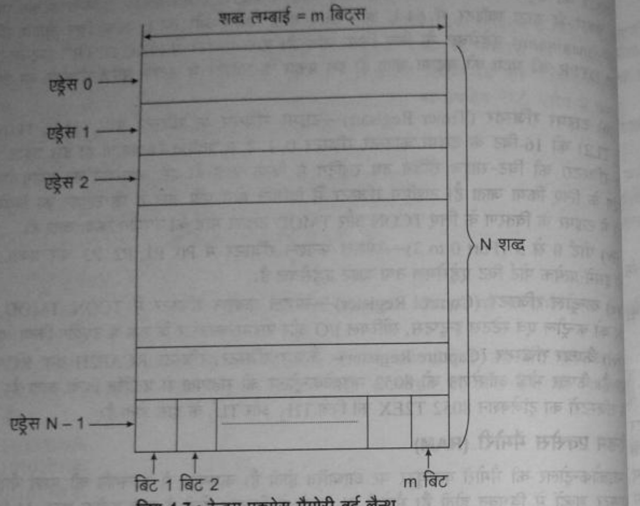
**(i) बाइपोलर मैमोरी** (Bipolar Memory)—इसमें मैमोरी सेल फ्लिप-फ्लॉप से बनी होती है। प्रत्येक फ्लिप-फ्लॉप 0 अथवा 1 बिट को स्टोर करता है। इस प्रकार की मैमोरी को फ्लिप-फ्लॉप तथा $P$-$N$ जेक्शन को मिलाकर बनाया जाता है।

**(ii) स्थैतिक मॉस मैमोरी** (Dynamic MOS Memory)—इसमें फ्लिप-फ्लॉप को MOS फील्ड इफैक्ट ट्रांजिस्टर से बनाया जाता है। यह बाईपोलर मैमोरी से कम गतिमान होती है।

**(iii) सीमोस मैमोरी** (CMOS Memory)—इस प्रकार की मैमोरी की निर्माण प्रक्रिया जटिल होती है। सीमोस मैमोरी $P$ तथा $N$ चैनल उपकरणों के प्रयोग से बनाई जाती है।

**(iv) सिलिकॉन ऑन सैफायर मैमोरी** (Silicon On Sapphire Memory)—सिलिकॉन ऑन सैफायर मैमोरी, सीमोस मैमोरी की तरह होती है। यह अन्य मैमोरियों से मँहगी होती है।

अत: इस प्रकार यह कहा जा सकता है माइक्रोकन्ट्रोलर में विभिन्न प्रकार के डाटाओं को स्टोर करने के RAM मैमोरी का ही उपयेाग किया जाता है। RAM मैमोरी का उपयोग इसलिए किया जाता है क्योंकि यह एक स्थायी मैमोरी है।



चित्र 4.7 : रेन्डम एक्सेस मैमोरी वर्ड लैन्थ

## माइक्रोकन्ट्रोलर में बिट एड्रेसेबल (Bit Addressable in Microcontroller)

माइक्रोकन्ट्रोलर में बिट एड्रेसेबल लोकेशन पूर्णत: RAM मैमोरी पर आधारित होता है। बिट एड्रेसेबल लोके किसी भी माइक्रोकन्ट्रोलर की प्रारम्भिक एवं अन्तिम स्थिति को दर्शाता है। बिट एड्रेस जनरल पर्पज RAM पर करता है। इस प्रकार के लोकेशन में विभिन्न प्रकार के स्पेशल फंक्शन रजिस्टरों का उपयोग किया जाता है जो वि प्रकार की बिटों को प्रदर्शित करता है।

माइक्रोकन्ट्रोलर में 210 बिट एड्रेसेबल लोकेशन होते हैं जिसमें 128 बिट 20 H से 2FH लोकेशन पर होते इस प्रकार के लोकेशन को स्पेशल फंक्शन रजिस्टर में रीसेट किया जाता है। इसमें प्रत्येक 128 बिटों को 20H से 2 में जोड़ने के लिए यूनिक नम्बर का प्रयोग किया जाता है। इस प्रकार से प्रत्येक इन्स्ट्रक्शन को सेट और रीसेट करने लिए RAM में एक सिंगल बिट का सिलेक्शन किया जाता है।

बिट एड्रेसेबल लोकेशन में जनरल पर्पज RAM को 30H से 7FH में तथा उसके रजिस्टर बैंक को 00H 1FH में दर्शाया जाता है। बिट लोकेशन में बाइट की स्थिति को लिखा व पढ़ा जा सकता है। बिट एड्रेसेबल RAM सिंगल बिट को पढ़ने व लिखने के लिए यूनिट एड्रेस का उपयोग किया जाता है। इस प्रकार बिट एड्रेसेबल लोकेशन उपयोग विभिन्न फंक्शनों में किया जाता है।

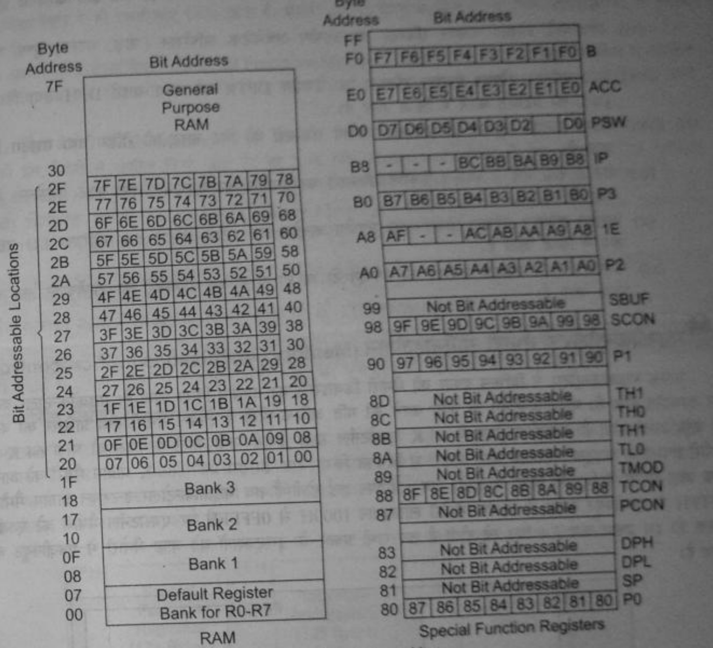
**उदाहरण**—एक सिंगल इन्स्ट्रक्शन SETB 67H को दर्शाने के लिए बिट एड्रेस 67H को HIGH लॉजिक किया जाता है तो 67H की बिट 7 को बाइट लोकेशन 2CH पर दर्शाता है। इस प्रकार माइक्रोप्रोसेसर निम्नलि फंक्शन दर्शाता है—

MOV A, 2CH

ORLA, # 10000000 B



MOV 2CH, A



चित्र 4.8 : बिट एड्रेसेबल लोकेशन

## माइक्रोकन्ट्रोलर के स्पेशल फंक्शन रजिस्टरों के उपयोग (Uses of Special Function Registers of Microcontroller)

माइक्रोकन्ट्रोलर के सभी स्पेशल फंक्शन रजिस्टरों के उपयोग निम्न हैं—

(i) प्रोग्राम स्टेटस वर्ड स्पेशल फंक्शन रजिस्टर में कैरी, ऑक्जिलियरी, ऑवरफ्लो और पेरटी फ्लैग आदि में $RS_0$ तथा $RS_1$ रजिस्टर का उपयोग करते हैं।

(ii) कैरी फ्लैग स्पेशल फंक्शन रजिस्टर का उपयोग अर्थमेटिक एवं लाजिकल क्रियाओं के ऑपरेशनों को परफॉर्म करने के लिए किया जाता है।

(iii) ऑक्जिलियरी कैरी फ्लैग स्पेशल फंक्शन रजिस्टर का उपयोग किसी फंक्शन की कैरी ज्ञात करने के लिए BCD अर्थमैटिक ऑपरेशन परफॉर्म किया जाता है।

(iv) F0 स्पेशल फंक्शन रजिस्टर का उपयोग सॉफ्टवेयर की सहायता से किसी रजिस्टर की स्थिति का पता लगाने में किया जाता है।

(v) *RS1* तथा *RS0* स्पेशल फंक्शन रजिस्टर का उपयोग किसी रजिस्टर की रेन्ज ज्ञात करने में ज्ञात जाता है।

(vi) ओवरफ्लो स्पेशल फंक्शन रजिस्टर का उपयोग अर्थमेटिक ऑपरेशन (जोड़, घटाना, गुणा, भा किया जाता है।

(vii) डाटा प्वॉइंटर स्पेशल फंक्शन रजिस्टर का उपयोग DPTR की उच्च बाइट DPH तथा निम्न DPL को प्रदर्शित करने मे किया जाता है।

(viii) टाइमर स्पेशल फंक्शन रजिस्टर का उपयोग रजिस्टरों की बिट-बाइट की रीडिंग तथा राइटिंग में जाता है।

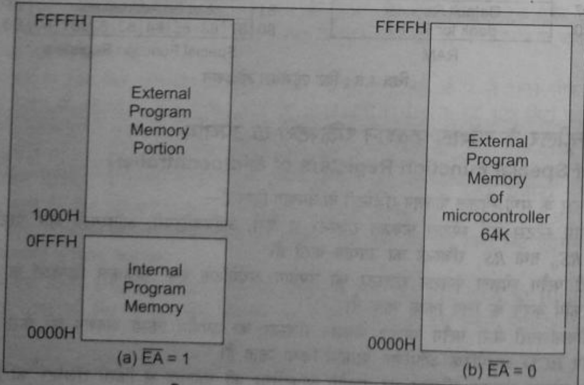
(ix) पोर्ट 0 तथा पोर्ट 3 स्पेशल फंक्शन रजिस्टरों का उपयोग बिट एड्रेसेबल तथा बाइट एड्रेसेबल में जाता है।

(x) कन्ट्रोल रजिस्टर स्पेशल फंक्शन का उपयोग कन्ट्रोल एवं स्टेटस इन्ट्रप्ट तथा सीरियल I/O टाइम रूप में किया जाता है।

(xi) कैप्चर रजिस्टर स्पेशल फंक्शन का उपयोग दो या दो से अधिक रजिस्टरों का ट्रान्जेक्शन ज्ञात क किया जाता है।

## माइक्रोकन्ट्रोलर में मैमोरी ऑर्गेनाइजेशन (Memory Organisation in Microcontroll

प्रत्येक माइक्रोकन्ट्रोलर में विभिन्न प्रकार की मैमोरी डिवाइसेस का उपयोग किया जाता है। माइक्रोकन्ट्रोलर कम पर आधारित होता है। माइक्रोकन्ट्रोलर के कार्य करने की गति अधिक होती है लेकिन मैमोरी तथा प्रोसेसर की का में बहुत अन्तर होता है। माइक्रोकन्ट्रोलर में 64 K एक्सटर्नल डाटा मैमोरी पायी जाती है, जिसे दो भागों 64 K प्रे मैमोरी तथा 256 बाइट्स आन्तरिक डाटा मैमोरी में विभक्त किया जाता है। इस प्रकार 64 K प्रोग्राम मैमेरी को आन तथा बाह्य मैमोरी में विभक्त किया जाता है। जब $\overline{EA}$ पिन हाई होती है तब माइक्रोकन्ट्रोलर इन्टरनल प्रोग्राम मैमों 0FFFH में एक्जीक्यूट करता है, जबकि मैमोरी लोकेश्यान 1000H से 0FFFFH पर एक्सटर्नल मैमोरी को एक्जी करता है। इस प्रकार जब *EA* पिन लो होती है तब सभी प्रकार के इन्स्ट्रक्शनों को बाह्य मैमोरी में एक्जीक्यूट क जाता है।




चित्र 4.9 : प्रोग्राम मैमोरी ऑर्गेनाइजेशन



अतः इस प्रकार के मैमोरी ऑर्गेनाइजेशन से यह कहा जा सकता है। सभी प्रकार के फंक्शनों तथ इन्स्ट्रक्शनों को मैमोरी ऑर्गेनाइजेशन में ही एक्जीक्यूट किया जाता है। मैमोरी ऑर्गेनाइजेशन के बिन्दुओं को निम्नलिखित प्रकार से समझा जा सकता है—

(i) **आन्तरिक प्रोसेसर मैमोरी** (Internal Processor Memory)—निर्देशों तथा डाटा को अस्थायी रूप से संग्रहित करने के लिए तीव्र गति रजिस्टरों का उपयोग किया जाता है। इन रजिस्टरों को ही आन्तरिक प्रोसेसर मैमोरी कहा जाता है।

(ii) **मुख्य मैमोरी** (Main Memory)—इसे प्राथमिक मैमोरी भी कहते हैं। कम्प्यूटर संक्रिया के समय डाटा तथा निर्देशों को इस मैमोरी में संग्रहित किया जाता है। यह उच्च गति (high speed) मैमोरी है तथा उसकी स्टोरेज कैपेसिटी (storage capacity) कम होती है। यह महँगी होती है। सेमीकण्डक्टर मैमोरी ROM, RAM, EPROM इसके उदाहरण हैं।

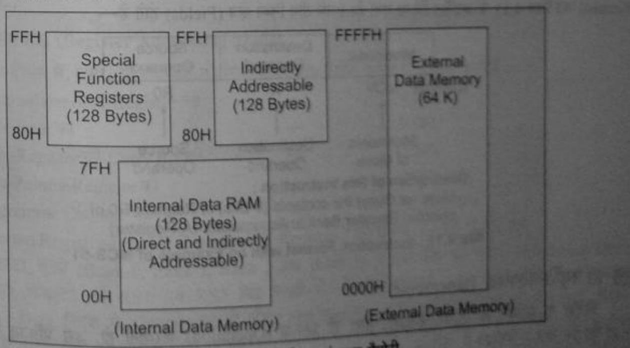
(iii) **द्वितीयक मैमोरी** (Secondary/Auxiliary Memory)—इसकी स्टोरेज कैपेसिटी (storage capacity) अधिक तथा गति कम होती है फ्लॉपी, हार्ड डिस्क, मैग्नेटिक टेप आदि इसके उदाहरण हैं।

## माइक्रोकन्ट्रोलर में आन्तरिक एवं बाह्य मैमोरी (Internal and External Memory in Microcontroller)

मैमोरी, माइक्रोकन्ट्रोलर का एक महत्त्वपूर्ण अंग है जिसमें विभिन्न प्रकार के फंक्शनों की सूचनाओं को स्टोर किया जाता है। मैमोरी से प्रोग्राम, डाटा, रिजल्ट तथा अन्य प्रकार की सूचनाओं की जानकारी प्राप्त की जाती है।

### आन्तरिक मैमोरी (Internal Memory)

माइक्रोकन्ट्रोलर की आन्तरिक मैमोरी में 256 बाइट होते हैं जो विभिन्न भागों में बँटे होते हैं। आन्तरिक मैमोरी के निचले भाग में 128 बाइट होते हैं जो 00H से 7FH मैमोरी लोकेशन तक पाये जाते हैं, जिन्हें आन्तरिक डाटा रैम (RAM) कहते हैं। इसी प्रकार इसके ऊपरी भाग में 128 बाइट होते हैं जो 80H से FFH मैमोरी लोकेशन में पाये जाते हैं, उन्हें स्पेशल फंक्शन रजिस्टर कहते हैं।




चित्र 4.10 : आन्तरिक एवं बाह्य मैमोरी

इस प्रकार की आन्तरिक मैमोरी में 8-बिट एड्रेस RAM मैमोरी, 16-बिट एड्रेस RAM प्रोग्राम मैमोरी तथा 8-बिट स्पेसीफिक RAM पर कार्य किया जाता है।

माइक्रोकन्ट्रोलर में मैमोरी के द्वारा सीधे जानकारी प्राप्त की जाती है। आन्तरिक मैमोरी विभिन्न अर्द्धचाल हिस्सा है इसलिए इसे सेमीकण्डक्टर मैमारी भी कहते हैं। आन्तरिक मैमोरी विभिन्न प्रकार के छोटे-छोटे मिलकर बनी होती है जिन्हें लोकेशन या सेल कहते हैं। मैमोरी में लोकेशन या वर्ड को एक नम्बर असाइन किया जो एड्रेस कहलाता है। आन्तरिक मैमोरी में मुख्यत: रेन्डंम एक्सेस मैमोरी का उपयोग किया जाता है जो माइक्रोकन्ट्रोलर की गति का निर्धारण करती है। रेन्डम एक्सेस मैमोरी विभिन्न साइजों में पायी जाती है। जैसे— 2MB, 4MB, 8MB, 16MB, 32MB, 64MB, 128MB आदि।

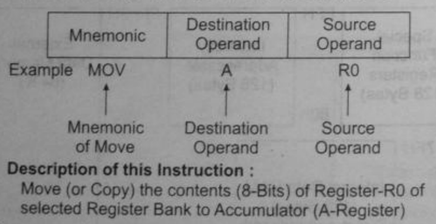
## बाह्य मैमोरी (External Memory)

माइक्रोकन्ट्रोलर की बाह्य मैमोरी 64 k बाइट के रूप में कार्य करती है। इस मैमोरी को ऑक्जिलरी मैमोरी से भी जाना जाता है। बाह्य मैमोरी के एड्रेस को 16-बिट एड्रेस मैमोरी तथा 16-बिट एड्रेस प्रोग्राम मैमोरी प्रदर्शित किया जाता है। यह मैमोरी नॉन-वोलेटाइल होती है जिसमें एक बार सूचना स्टोर करने के बाद उसका कभी भी किया जा सकता है, एवं उसे कभी भी प्रोसेस करवाया जा सकता है। इस प्रकार की मैमोरी का उपयोग सिस्टम में बड़ी-बड़ी डाटा फाइलों को संग्रहित करने में किया जाता है। इस मैमोरी का उपयोग माइक्रोप्रोसेसर नहीं करता है। यह मैमोरी आन्तरिक मैमोरी से अधिक स्पीड वाली होती है।

इस प्रकार यह कहा जा सकता है कि यह मैमोरी एंक स्थायी मैमोरी है जिसका उपयोग प्राथमिक एवं दोनों प्रकार की मैमोरियों में किया जाता है। बाह्य मैमोरी काफी महँगी होने के कारण अधिक स्टोरेज कैपेसिटी उपयोग में लायी जाती है। फ्लॉपी डिस्क, हार्ड डिस्क, मैग्नेटिक डिस्क, मैग्नेटिक टेप और पेन ड्राइव आदि बाह्य के उदाहरण हैं।

## MCS-51 फैमली के माइक्रो-कन्ट्रोलरों के एड्रैसिंग मोड (Addressing Modes of MCS-51 Family Micro-controllers)

माइक्रो-कन्ट्रोलरों/माइक्रो-प्रोसेसरों में उपयोग किये जाने वाले एक सामान्य निर्देश के प्रकार (Instruction Format) को चित्र 4.11 में प्रदर्शित किया गया है। इसमें तीन निम्न क्षेत्र (Fields) होते हैं—




चित्र 4.11 : Instruction Format with an Example of MCS-51

### (i) स्मृति-सहायक (Mnemonic)

निर्देश के माध्यम से माइक्रो-कन्ट्रोलर द्वारा जो प्रक्रिया (Operation) करवानी हो, उस प्रक्रिया के संकेत शब्द को इस क्षेत्र में लिखिते हैं, जैसे—

Move के लिए "MOV";

Addition के लिए "ADD",

Exchange के लिए "XCH" आदि का प्रयोग निर्देशों में किया जाता है।



### (ii) गन्तव्य/लक्ष्य ऑपरेण्ड (Destination Operand)

यहाँ पर SFR/Memory Location/PC के संकेत/संक्षेप-अक्षर को अंकित करते है जो प्रभावी होता है अर्थात् जिसके Contents में परिवर्तन Mnemonic तथा Source Operand के अनुसार होता है। उदाहरणतया—

MOV A, #57H : Move (Store) 57H (Hexa-decimal) Data in Accumulator

इसमें *A*-रजिस्टर अर्थात् एक्यूमुलेटर, इस निर्देश में Destination Operand है।

### (iii) स्रोत ऑपरेण्ड (Source Operand)

इस क्षेत्र में उस SFR/Memory Location/Direct Data को अंकित करते हैं, जिसके Contents को Destination Operand में स्टोर करना हो। उपरोक्त उदाहरण में "#57H" डाटा Source Operand है। इसी प्रकार निम्न उदाहरण में—

MOV R1, A : Move Accumulator Contents to R1-Register of Selected Bank

MOV ⇒ Move प्रक्रिया का Mnemonic है

R1 ⇒ इस निर्देश का Destination Operand तथा

*A* ⇒ इस निर्देश का Source Operand है।

MCS-51 में किसी मैमोरी स्थान (Memory Location), Special Function Register (SFR), अन्य रजिस्टर आदि को Addressing करने के निम्न पाँच एड्रैसिंग मोड (Addressing Modes) होते हैं—

- रजिस्टर एड्रैसिंग मोड (Register Addressing Mode)
- प्रत्यक्ष एड्रैसिंग मोड (Direct Addressing Mode)
- रजिस्टर अप्रत्यक्ष एड्रैसिंग मोड (Register-Indirect Addressing mode)
- तात्कालिक डाटा एड्रैसिंग मोड (Immediate Data Addressing or Immediate Addressing Mode)
- आधार रजिस्टर + सूचक रजिस्टर एड्रैसिंग मोड (Base Register Plus Index Register Addressing Mode)

**रजिस्टर एड्रैसिंग मोड (Register Addressing Mode)**

रजिस्टर एड्रैसिंग में, CPU के रजिस्टरों को निम्न संकेत द्वारा निर्देश-समूह में अंकित करते हैं—

| | | |
|---|---|---|
| Accumulator or A-Register को | — | *A* द्वारा, |
| B-Register को | — | *B* द्वारा, |
| PSW-Register की Carry Bit को | — | *C* द्वारा, |
| Data-Pointer Register को | — | DPTR द्वारा, |
| Programme Counter को | — | PC द्वारा, तथा |

Selected Register-Bank के Register-0 से Register-7 को क्रमश: — R0 से R7 द्वारा

चार रजिस्टर बैंकों (Bank-0, Bank-1, Bank-2 तथा Bank-3) में एक बैंक का चयन, Programme Status World (PSW) रजिस्टर की RS1 एवं RS2 बिटें करती है, जैसाकि तालिका 6.6.3 में प्रदर्शित किया गया है। MCS-51 के 1-Byte निर्देश में, Opcode की ही तीन Least Significant Bits (LSBs : B0, B1 एवं B2), चयनित रजिस्टर बैंक के 8-रजिस्टरों (R0 से R7) में से एक रजिस्टर का चयन करती है तथा अन्य 5-Bits निर्देश के Opcode को दर्शाती हैं। चयनित (Selected) रजिस्टर-बैंक के रजिस्टर (R0 से R7 तक में से एक) के उपयोग द्वारा सम्बोधित एक निर्देश-फॉर्मेट (Instruction Format) को चित्र 4.12 में प्रदर्शित किया गया है।

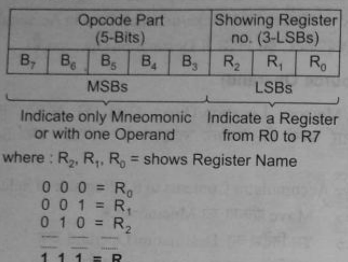
रजिस्टर-एड्रैसिंग मोड के मुख्य उदाहरण निम्न हैं—

(i) MOV R5, A : Move the Contents of Accumulator to the R5-Register of Selected Bank

इस निर्देश का Machine Language में Code जिसे Machine Code कहते हैं, निम्न होता है—

1 1 1 1 1    1 0 1    अर्थात्    FDH

Opcode   Part    Register No.




**चित्र 4.12 : Instruction Format of Register Addressing Mode in Instruction of MCS-51**

अत: इस निर्देश का अर्थ है—

$$(A) \Rightarrow (R5)$$

जहाँ, $(A)$ = Contents of $A$-Register (Accumulator)

$(R5)$ = Contents of R5-Register of Selected Bank

(ii) INC R7 : Increment the Contents of Register – R7 of Selected Bank

इस निर्देश का Machine Code निम्न होता है—

0 0 0 0 1    1 1 1    अर्थात्    OFH

Mnemonic    Register No.

here $\Rightarrow R7$

इस निर्देश का अर्थ है—

$$(R7) + 1 \Rightarrow (R7)$$

अर्थात् $R7$-रजिस्टर के Contents में 1-अंक की वृद्धि करके, प्राप्त Contents को पुन: $R7$-रजिस्टर मे करना।

इस प्रकार के Register Addressing Mode में प्रदर्शित निर्देश को **रजिस्टर-निर्देश** (Register Instructi भी कहते हैं।

## प्रत्यक्ष एड्रैसिंग मोड (Direct Addressing Mode)

इस मोड में, ऑपरेण्ड (Operand) के मैमोरी स्थान (8-Bit Address) अथवा Special Function Reg (SFR) को निर्देश में अंकित किया जाता है। इस प्रकार निर्देशों में केवल आन्तरिक RAM की Lower 128-Byt Address का अथवा SFR के विशेष संकेत का प्रयोग किया जाता है। इस प्रकार के निर्देश के फॉर्मेट को चित्र 4.1 प्रदर्शित किया गया है।

प्रत्यक्ष एड्रैसिंग मोड के कुछ उदाहरण निम्न हैं—

(i) MOV A, Direct

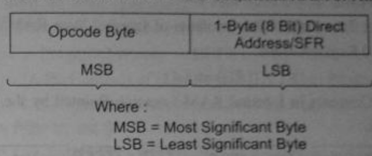
(ii) MOV Direct, Rn (जहाँ, $n = 0$ से 7 तक कोई एक संख्या, जैसे $R5$, $R2$ इत्यादि)



(iii) ADD A, Direct

(iv) DEC Direct

जहाँ, Direct = Any One Address from 00H to 7FH of Internal RAM or Any One SFR




**चित्र 4.13 : Instruction Format of Direct Addressing Mode in Instruction of MCS-51**

अत: निम्न-प्रदर्शित निर्देशों का निम्न अर्थ है—

**(i) MOV A, 7BH :** Move contents of 7BH Location of Internal RAM to Accumulator

अर्थात्    (7BH) ⇒ (A)

जहाँ, (7BH) का अर्थ है = Contents of 7BH-Location of Internal RAM

(A) = Contents of Accumulator

**(ii) MOV 57H, A :** Move Contents of Accumulator to the Register of Internal RAM at Location 57H

अर्थात्    (A) ⇒ (57H)

**(iii) ADD A, 27H :** Add the contents of Internal RAM Location 27H with the contents of Accumulator and Result store in Accumulator.

अर्थात्    (A) + (27H) ⇒ (A)

**(iv) MOV A, SBUF :** Move the contents of SBUF (Serial Data Buffer) Register to Accumulator जहाँ; SBUF एक SFR है।

अर्थात्    (SBUF) ⇒ (A)

**(v) DEC 61H :** Decrement the contents of Register which located at 61H Address of Internal RAM

अर्थात्    (61H) – 1 ⇒ (61H)

## रजिस्टर अप्रत्यक्ष एड्रैसिंग मोड (Register Indirect Addressing Mode)

PSW रजिस्टर की RS1 एवं RS0 बिटों के संयोग से चयनित (selected) रजिस्टर बैंक, जिसके चयन (select) करने की विधि को तालिका 7.6.3 में प्रदर्शित किया गया है, के $R0$ अथवा $R1$ रजिस्टर को अथवा डाटा-संकेतक (Data Pointer; DPTR) को आन्तरिक एवं बाह्य तीनों RAMs के अलग-अलग 256-Bytes Location के लिए संकेतक (Pointer) के रूप में उपयोग किया जाता है। इस प्रकार, आन्तरिक RAM की Lower 128-Bytes (00H से 7FH Locations तक) एवं केवल 8032/52 माइक्रो-कन्ट्रोलरों में आन्तरिक Ram की upper 128-Bytes (80H से FFH Location तक) को संकेत (Point) किया जा सकता है, जैसा कि चित्र 4.14 (a) में प्रदर्शित किया गया है अथवा बाह्य डाटा मैमोरी अर्थात् External RAM के केवल Lower 256-Bytes (00H से FFH Locations तक) को संकेत (Point) कर सकते हैं, जैसा कि चित्र 4.14 (b) में प्रदर्शित किया गया है।

➤ नोट-इस मोड द्वारा Special Function Registers (SFRs) तथा बाह्य RAM के Lower 256-Byte Locations से ऊपर वाली Locations (0100H से FFFFH तक) को Address नहीं किया जा सकता है

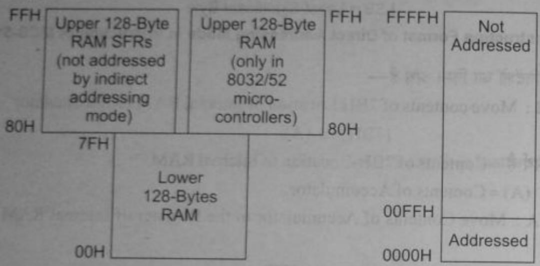
परन्तु डाटा संकेतक (Data Pointer-DPTR), जोकि 16-Bit रजिस्टर है, के द्वारा बाह्य RAM के साथ 64k-Byte Locations (0000 H) से FFFFH तक) को Address किया जा सकता है।

रजिस्टर अप्रत्यक्ष एड्रैसिंग मोड के कुछ निर्देश निम्न है—

**(i) MOV A, @ R1 :** Move the 8-Bit Contents of Internal Data RAM Location Pointed by Contents of R1 of Selected Register-Bank

अर्थात् $((R1)) \Rightarrow (A)$

जहाँ, $((R1))$ = Store Contents in Internal RAM Location Pointed by the Contents of R1-Register of Selected Register-Bank.




(a) : Lower 128-Bytes Internal RAM Locations + are Addressed by the Register Indirect Addressing Mode

(b) : Only Lower 256-Bytes of External RAM Locations are Addressed by the Register Indirect Addressing Mode

Upper 128-Bytes Internal RAM (only in 8032/52 Micro-Controllers)

चित्र 4.14

(A) = Contents of Accumulator

**(ii) MOV @ R0, A :** Move the Contents of Accumulator to **Internal RAM Location** Pointed by the Register R0 of Selected Register Bank.

अर्थात् $(A) \Rightarrow ((R0))_{Internal}$

**(iii) MOV X @ R1, A :** Move the Contents of Accumulator to **External RAM** Location Pointed by Register R1 of Selected Register Bank.

अर्थात् $(A) \Rightarrow ((R1))_{External}$

### तात्कालिक डाटा अथवा स्थिर एड्रैसिंग (Immediate Data or Constants Addressing)

इसमें तात्कालिक डाटा (अर्थात् Constants) निर्देश का ही एक भाग होता है। उदाहरणतया निम्न निर्देशों में—

(i) MOV A, #75H : Move (Store) the Immediate Data (Constants) – 75H in the Accumulator

अर्थात् 75H ⇒ (A)

(ii) MOV A, #105 : Move (Store) the Decimal Number (Immediate Data or Constants) 105 in the Accumulator

अर्थात् Decimal Number 105 ⇒ (A)



### (इण्डैक्स रजिस्टर + बेस रजिस्टर) – अप्रत्यक्ष एड्रैसिंग मोड (Index Register Plus Base Register – Indirect Addressing Mode or Indexed Addressing Mode)

इस मोड द्वारा प्रोग्राम मैमोरी की स्थिति (Location) में स्टोर 1-Byte को केवल पढ़ने (Read) के लिए एक्सेस (Access) किया जा सकता है तथा Location के Address की गणना निम्न प्रकार से की जाती है—

Address of Location in Program Memory = (DPTR or PC) + (A)

जहाँ, (DPTR or PC) = 16-Bit Contents of Data-Pointer (DPTR) or Programme Counter (PC)

(A) = 8-Bit Contents of Accumulator (Index Register)

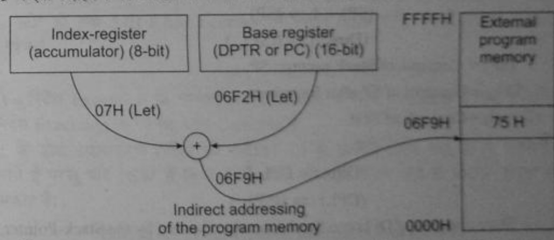
उदाहरण के लिए, निम्न निर्देश का अर्थ है—

**MOV A, @ A+PC :** Move (Store) a Byte (8-Bit) in Accumulator from the Location of Program Memory, whose Address is calculated by Adding 8-Bit (one-Byte) unsigned Contents of the Accumulator and the 16-Bit (2-Byte) Contents of PC.

माना PC ने 16-Bit Contents-06F2H स्टोर है तथा Accumulator (A-Register) में 8-Bit Contents-07H स्टोर है तो ये संयुक्त रूप से निम्न Address को संकेत (Point) करते है—

06F2H = Contents of PC
+ 07H = Contents of Accumulator
06F9H = Address of Programme Memory Location

चित्र 4.15 में इस उदाहरण को आरेख के माध्यम से प्रदर्शित किया गया है।




चित्र 4.15 : Index register plus base register – indirect addressing mode (or indexed addressing mode)

CPU द्वारा इस निर्देश (Instruction) के पश्चात् प्रोग्राम मैमोरी के Location-06F9H में स्टोर 8-Bit (one-Byte) Contents-75H की कापी (Copy) Accumulator (A-रजिस्टर) में हो जाती है।

## MCS-51 के निर्देश-प्रकार (Instruction Types of MCS-51)

MCS-51 के 111-निर्देश होते है जिनको निम्न प्रमुख वर्गों (Groups) में विभाजित कर सकते है—

- डाटा स्थानान्तरण निर्देश (Data Transfer Instructions)
- गणितीय निर्देश (Arithmetic Instructions)
- लॉजिक निर्देश (Logical Instructions)
- बिट प्रचालित निर्देश (Bit Operating Instructions)

- प्रोग्राम शाखन निर्देशन (Programme Branching Instructions)

## MCS-51 के डाटा स्थानान्तरण निर्देश (Data Transfer Instructions of MCS-51)

MCS-51 के निर्देशों में डाटा स्थानान्तरण में निम्न स्मृति-सहायकों (Mnemonics) का उपयोग किया जाता है—

**(i) MOV :** Special Function Registers (SFRs) के मध्य परस्पर डाटा-स्थानान्तरण के लिए तथा आ... RAM (अर्थात् Data Memory) एवं SFRs के मध्य परस्पर डाटा-स्थानान्तरण के लिए निर्देश में "M... स्मृति-सहायक संकेत प्रयोग किया जाता है।

**(ii) MOVC :** 16-Bit Addressable बाह्य प्रोग्राम मैमोरी से 1-Byte (8-Bit) Code को एक्यूमुलेटर (A... स्थानान्तरण अर्थात् कॉपी करने के लिए MOVC स्मृति-सहायक संकेत प्रयोग किया जाता है।

**(iii) MOVX :** बाह्य डाटा RAM एवं आन्तरिक SFRs में परस्पर डाटा स्थानान्तरण अर्थात् कापी (C... करने के लिए "MOVX" स्मृति-सहायक संकेत का प्रयोग किया जाता है।

**(iv) PUSH :** CPU में PUSH निर्देश प्राप्त (Fetch) होते ही Stack Pointer (SP) में स्टोर 8-Bit Con... में '1' की वृद्धि हो जाती है और अब SP में स्टोर यह न्यू 8-Bit, आन्तरिक RAM के जिस Stack Locatio... संकेत करती है उसमे निर्देश के साथ अंकित Operand के स्थान पर SFR अथवा RAM Location के Con... स्टोर हो जाते हैं।

**उदाहरण**—निम्न PUSH निर्देशन निम्न प्रकार से कार्य करता है—

**PUSH DPL :** Push (Store) Low-Byte of the Data Pointer (DPTR) onto Stack Location poi... by Stack Pointer

$$(SP) + 1 \Rightarrow (SP)_n$$
$$(Direct) \Rightarrow ((SP)_n)$$

जहाँ, $(SP)$ = Contents of Stack-pointer; SP

$(SP)_n$ = Contents of SP after Increment

$(Direct)$ = Contents of SFR

इस निर्देश में—

$$Direct = DPL \text{ है}$$
$$(DPL) \Rightarrow ((SP)_n)$$

$((SP)_n)$ = Store contents of DPl onto Stack Location Pointed by the Stack-Pointer.

**(v) POP : RAM** के Stack में से 1-Byte को बाहर निकालने के POP निर्देश का प्रयोग करते हैं तथा... POP Operation भी कहा जाता है।

Stack मैमोरी एक LIFO संरचना की भाँति कार्य करती है अत: Stack में अन्त में स्टोर की गई 1-By... Information पहले (First) बाहर (Out) आती है। उदाहरणतया आगे दिया POP निर्देश के Execution में नि... प्रक्रिया होती है—

**POP DPL :** Pop the Contents of the Stack Location Pointed by SP onto the DPL-Register अर्थात् संक्षेप में निम्न प्रक्रिया होती है—

$$((SP)) \Rightarrow (Direct) \text{ यहाँ } Direct = DPL, \text{ अत:}$$
$$((SP)) \Rightarrow (DPL)$$

तथा

$$(SP) - 1 \Rightarrow (SP)_n$$

जहाँ, $((SP))$ = SP द्वारा संकेतित (pointed) Stack Location से DPL-रजिस्टर में Contents स्टोर होना।

$(SP)$ = SP में स्टोर Contents



$(SP)_n$ = Decrement के पश्चात् SP में स्टोर Contents

**(vi) XCH :** इस निर्देश का प्रयोग Accumulator तथा किसी अन्य SFR अथवा आन्तरिक RAM मैमोरी के रजिस्टर के Contents की परस्पर अदला-बदली (Exchange) के लिए किया जाता है। उदाहरणतया—यदि R5 में स्टोर 57H हो तथा Accumulator (A) में ACH हो तो XCH A, R5 निर्देश के Execution के पश्चात् (A) एवं (R5) के मान निम्न होंगे—

$$(A) = 57H \text{ तथा } (R5) = ACH$$

**(vii) XCHD :** Exchange Digit

***1. स्टैक** (Stack)—Stack, आन्तरिक RAM में उन Memory Locations का समूह है जो Program Execution के समय Binary Informat (or Data) को अस्थायी (Temporary) रूप से संचित (Store) करने के लिए प्रयोग किया जाता है। Stack का स्थान (Locations) RAM में आरक्षित होता है। प्राय: Stack स्थान, Memory-Map के High End पर होता है।

***2. LIFO Structure of Stack**—यह Last-In First-Out का संक्षिप्त रूप है। LIFO जिसका अर्थ है कि Last में स्टोर की जाने वाली Information (यहाँ 1-Byte), पहले (First) बाहर (Out) आती है। Stack भी इसी प्रकार की मैमोरी होती है, अर्थात् Stack Memory में कोई Informulation (Data) को Top Location में ही PUSH निर्देश द्वारा Add कर सकते हैं तथा Top से ही उसे POP निर्देश द्वारा प्राप्त कर सकते हैं।

इस निर्देश का प्रयोग, Accumulator की Lower Order Nibvle (Bits 3-0) एवं आन्तरिक RAM Location की Lower Order Nibble को परस्पर अदला-बदली (Exchange) करने के लिए प्रयोग करते हैं। दोनों रजिस्टरों की High Order Nibbles (Bits 7-4) में कोई परिवर्तन नहीं होता है। उदाहरणतया—यदि (A) = 29H हो एवं R1 में 57H Address स्टोर हो तथा 57H RAM Location में 9AH Data स्टोर हो तो निर्देश XCHDA, @ R1 के Execution के पश्चात्

$$(A) = 9AH \text{ तथा } (57H) = 29H$$

जहाँ, (A) = निर्देश Execution के पश्चात् Accumulator में स्टोर Contents है तथा (57H) = 57H RAM Location में निर्देश Execution के पश्चात् स्टोर Contents है।

MCS-51 के डाटा-स्थानान्तरण निर्देशों को तालिका 4.1 में प्रदर्शित किया गया है। ये निर्देश PSW Flags को प्रभावित नहीं करते हैं परन्तु यदि PSW में किसी Contents को स्टोर करना चाहे तो MOV अथवा POP निर्देश द्वारा ऐसा किया जा सकता है।

**तालिका 4.1 : Data Transfer Instructions of MCS-51**

| Mnemonic | | Description | Bytes | OSC Period |
|---|---|---|---|---|
| MOV | A,Rn | Move register to Accumulator | 1 | 12 |
| MOV | A,direct | Move direct byte to ACC | 2 | 12 |
| MOV | A,@Ri | Move indirect RAM to ACC | 1 | 12 |
| MOV | A,#data | Move immediate data to ACC | 2 | 12 |
| MOV | Rn,A | Move accumulator to Register | 1 | 12 |
| MOV | Rn,direct | Move direct byte to Register | 2 | 24 |
| MOV | Rn,#data | Move immediate data to Register | 2 | 12 |
| MOV | direct,A | Move Accumulator to direct byte | 2 | 12 |
| MOV | direct,Rn | Move Register to direct byte | 2 | 24 |
| MOV | direct,direct | Move direct byte to direct byte | 3 | 24 |

| | | | |
|---|---|---|---|
| MOV direct,@Ri | Move indirect RAM to direct byte | 2 | 24 |
| MOV direct, # data | Move immediate data to direct byte | 3 | 24 |
| MOV @Ri,A | Move Accumulator to indirect RAM | 1 | 12 |
| MOV @Ri,direct | Move direct byte to indirect RAM | 2 | 24 |
| MOV @Ri,#data | Move immediate data to indirect RAM | 2 | 12 |
| MOV DPTR,#data 16 | Load Data Pointer with 16-bit Constant | 3 | 24 |
| MOVC A,@A+DPTR | Move Code byte relative to DPTR to Accumulator | 1 | 24 |
| MOVE A,@A+PC | Move Code byte relative to PC to Accumulator | 1 | 24 |
| MOVX A,@Ri | Move Ext. RAM (8-bit addr.) to Accumulator | 1 | 24 |
| MOVX A,@DPTR | Move Ext. RAM (16-bit addr) to Accumulator | 1 | 24 |
| MOVX @Ri,A | Move ACC to Ext. RAM (8-bit addr.) | 1 | 24 |
| MOVX @DPTR,A | Move ACC to Ext. RAM (16-bit Addr.) | 1 | 24 |
| PUSH direct | Push direct byte in Stack | 2 | 24 |
| POP direct | Pop direct byte from Stack | 2 | 24 |
| XCH A,Rn | Exchange Register with Accumulator | 1 | 12 |
| XCH A,direct | Exchange direct byte with Accumulator | 2 | 12 |
| XCH A,@Ri | Exchange indirect RAM with Accumulator | 1 | 12 |
| XCHD A,@Ri | Exchange low order digit indirect RAM with ACC | 1 | 12 |

direct = 8-Bit Internal RAM Location's Address. This could be an Internal Data RAM Locatio (00H-7FH) of Special Function Register (SFR).

Rn = A Register From R0 to R7 of the currently selected Register Bank.

#data = Immediate Data Byte (One Byte Data from 00H to FFH) specified in the Instruction.

#data 16 = 16-Bit (= 2-Byte) Immediate Data specified in the Instruction.

@Ri = 8-Bit (= 1-Byte) Internal Data RAM Location Address pointed by the Contents R0 or R which specified inthe Instruction.

A = Accumulator (ACC)

@DPTR = Address of External RAM (Data) Memory Location pointed by the contents of DPTR

@A+DPTR = Address of External Programme Memory Location which is Equal to the sum o Contents of DPTR and ACC.

@A+PC = 16-Bit Address of External Ram Memory Location which is Equal to the sum o Contents of PC and ACC.

## MCS-51 के गणितीय निर्देश (Arithmetic Instructions of MCS-51)

MCS-51 द्वारा चिन्ह रहित संख्याओं पर गणितीय प्रक्रियायें (जैसे—जोड़, घटा, गुणा, भाग आदि) करना सम्भव है। इसमें ओवरलोड (Overload) Flag का प्रयोग कर, चिन्हित एवं चिन्ह-रहित (Signed and unsigned) संख्याओं का जोड़ (Addition) एवं घटा (Subtraction) सम्भव है। MCS-51 द्वारा BCD संख्याओं पर BCD गणितीय प्रक्रियायें भी की जा सकती हैं। ये सभी प्रक्रियायें, निर्देशों द्वारा MCS-51 में की जाती हैं। MCS-51 के गणितीय निर्देशों को तालिका 4.2 में प्रदर्शित किया गया है—



तालिका 4.2 : Arithmetic Instruction of MCS-51

| Mnemonic | | Description | Bytes | OSC Period |
|---|---|---|---|---|
| ADD | A,Rn | Add Register to Accumulator | 1 | 12 |
| ADD | A,direct | Add direct byte to ACC | 2 | 12 |
| ADD | A,@Ri | Add indirect RAM to ACC | 1 | 12 |
| ADDC | A,#data | Add immediate data to ACC | 2 | 12 |
| ADDC | A,Rn | Add Register to ACC with Carry | 1 | 12 |
| ADDC | A,direct | Add direct byte to ACC with Carry | 2 | 12 |
| ADD | A,@Ri | Add indirect RAM to ACC with Carry | 1 | 12 |
| ADD | A,#data | Add immediate data to ACC with Carry | 2 | 12 |
| SUBB | A,Rn | Subtract Register fromACC with Borrow | 1 | 12 |
| SUBB | A,direct | Sub. direct byte from ACC with Borrow | 2 | 12 |
| SUBB | A,@Ri | Sub. indirect RAM fromACC with Borrow | 1 | 12 |
| SUBB | A,#data | Sub. imm. data from ACC Borrow | 2 | 12 |
| INC | A | Increment Accumulator | 1 | 12 |
| INC | Rn | Increment Register | 1 | 12 |
| INC | direct | Increment direct byte | 2 | 12 |
| INC | @Ri | Increment indirect RAM | 1 | 12 |
| DEC | A | Decrement Accumulator | 1 | 12 |
| DEC | Rn | Decrement Register | 1 | 12 |
| DEC | direct | Decrement direct byte | 2 | 12 |
| DEC | @Ri | Decrement indirect RAM | 1 | 12 |
| INC | DPTR | Increment Data Pointer | 1 | 24 |
| MUL | AB | Multiply A and B (16-bit Result in B-A registers) | 1 | 48 |
| DIV | AB | Divide A by B (quotient in A, registers in B) | 1 | 48 |
| DA | A | Decimal adjust Accumulator | 1 | 12 |

where, B = B-Register (a SFR)

## MCS-51 के लॉजिक निर्देश (Logic Instructions of MCS-51)

MCS-51 द्वारा बिट एवं बाइट (Bit and Byte) दोनों Operands पर बेसिक लॉजिक प्रक्रियाये (AND, OR, Exclusive-OR and NOT Operations) करना सम्भव है। लॉजिक निर्देशों में स्मृति-सहायक (Mnemonic) के पश्चात् एक अथवा दो Operand (s) होते हैं।

### (1) एकल ऑपरेण्ड युक्त निर्देश (Single-Operand Instructions)

MCS-51 के एकल Operand वाले लॉजिक निर्देश एवं उनकी प्रक्रियायें निम्न हैं—

**(i) CLR A :** Clear Accumulator

इसका प्रयोग Accumulator को Reset करने के लिए किया जाता है। इस निर्देश के Execution के पश्चात्

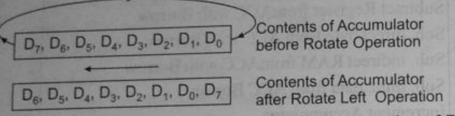
(A) = 00000000 अर्थात् = 00H

**(ii) CPL A :** Complement Accumulator

इस निर्देश के Execution के पश्चात्, Accumulator की सभी 8-बिट में से प्रत्येक बिट अपने से वि (Complement अथवा NOT) बिट द्वारा परिवर्तित हो जाती हैं। उदाहरणतया—यदि Accumulator में Contents -(A) = 01001001 है तो CPL A निर्देश के Execution के पश्चात् (A) = 10110110 हो जाता है।

**(iii) RL A :** Rotate the Contents of Accumulator to Left

इस प्रक्रिया को आरेख द्वारा चित्र 4.16 में प्रदर्शित किया गया है। इसमें सभी बिटें अग्र-बिट उच्च स्थान (N Bit High Space) पर शिफ्ट हो जाती है तथा 7th-बिट के Contents 0-बिट स्थान पर शिफ्ट हो जाते उदाहरणतया यदि (A) = 1 0 1 0 1 1 0 0 हो तो RLA निर्देश के Execution के पश्चात् (A) = 0 1 0 1 1 0 0 1 जाता है।

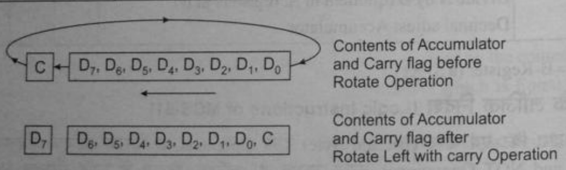



**चित्र 4.16 : Rotate Left Accumulation Operation or Illustration to excution of RLA Instruction**

**(iv) RLC A :** Rotate the Contents of Accumulator Left through Carry

इस प्रक्रिया को आरेख (Diagram) द्वारा चित्र 4.17 में प्रदर्शित किया गया है। इस निर्देश (RLC A) Execution से 7th बिट ($D_7$) Carry Flag स्थान पर पहुँच जाती है तथा Carry Flag बिट (c), 0-बिट पर शि हो जाती है। RLC A निर्देश के उदाहरण को नीचे प्रदर्शित किया गया है।

| Contents Before Operation | | Contents After RLC A Operation | |
|---|---|---|---|
| C | Accumulator | C | Accumulator |
| 1 | 01011000 | 0 | 10110001 |



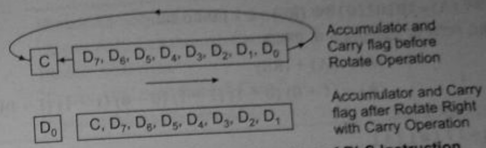

**चित्र 4.17 : Illustration of the Execution of RLC Instruction (Rotate Accumulator's Contents Left Through Carry Operation)**

➤ **नोट**–इसी प्रकार RR A (Rotate Right Accumulator) एवं RRC A (Rotate Right Accumulator Through Carry) निर्देश होते हैं। RRC A निर्देश को आरेख द्वारा चित्र 4.18 में प्रदर्शित किया गया है।

**(v) SWAP A :** Swap (Exchange) Nibbles (4-Bits) within the Accumulator

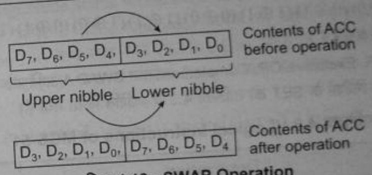
इस प्रक्रिया को आरेख द्वारा चित्र 4.19 में प्रदर्शित किया गया है। इस निर्देश का संख्यात्मक उदाहरण अग्र प्रदर्शित किया गया है।






**चित्र 4.18 : Illustration of the Execution of RLC Instruction (Rotate ACC's Contents Right Through Carry Operation)**

| Contents of Accumulator Before Operation | Contents of Accumulatoro After SWAP A Operation |
|---|---|
| 00001111 | 11110000 |




**चित्र 4.19 : SWAP Operation**

**(2) द्वि-ऑपरेण्ड युक्त निर्देश (Two-Operand Intructions)**

MCS-51 के द्वि-ऑपरेण्ड लॉजिक निर्देश एवं प्रक्रियायें निम्न हैं—

माना $OP_1$ = Ist Operand of Instruction

तथा $OP_2$ = IInd Operand of Instruction

**(i) ANL $OP_1, OP_2$ :** AND-Logic performs Bitwise of two Source Operands ($OP_1$ and $OP_2$) of 1-Byte Each and stores the Result to the Location of the Ist Operand

where; $OP_1$ = Accumulator (A) or Any Register of RAM

$OP_2$ = Accumulator of Any one Register from R0 to R7 of selected Register-Bank or Immediate Data (# Data)

अर्थात् $$(OP_1)\cdot(OP_2) \Rightarrow (OP_1)_n$$

जहाँ, • = Symbol of AND Logic Operation

उदाहरणतया—यदि (A) = 1 0 1 0 1 1 0 1 तथा (R0) = 0 1 1 0 1 0 1 0

तो ANL A, R0 निर्देश के Execution के पश्चात्

$$(A) = (1.0)\ (0.1)\ (1.1)\ (0.0)\ (1.1)\ (1.0)\ (0.1)\ (1.0)$$

जहाँ, • = AND Logic Symbol है।

अत: **(A) = 0 0 1 0 1 0 0 0**

**(ii) ORL $OP_1, OP_2$ :** Logical-OR performs Bitwise of Two Source Operands

अर्थात् $$(OP_1) + (OP_2) \rightarrow (OP_1)_n$$

उदाहरणतया—यदि (A) = 10101101 तथा (R0) = 0 1 1 0 1 0 1 0

हो तो ORLA, R0 निर्देश के Execution के पश्चात्

$$(A) = (A) + (R0)$$

अतः $(A) = (1 + 0)\,(0 + 1)\,(1 + 1)\,(0 + 0)\,(1 + 1)\,(1 + 0)\,(0 + 1)\,(1$

जहाँ, $+$ = OR-Logic

अर्थात् **(A) = 1 1 1 0 1 1 1 1**

**(iii) XRL $OP_1$, $OP_2$ :** Logical Exclusive-OR (XOR) performs Bitwise of Two Source Operand

अर्थात् $(OP_1) \oplus (OP_2) \Rightarrow (OP_1)_n$

जहाँ, $\oplus$ = XOR का लॉजिक चिन्ह

उदाहरणतया—यदि (A) = 10101101 तथा (R0) = 01101010 हो तो **XRL A, R0** निर्देश के Executio पश्चात्

$$(A) = (A) \oplus (R0)$$

अतः $(A) = (1 \oplus 0)(0 \oplus 1)(1 \oplus 1)(0 \oplus 0)(1 \oplus 1)(1 \oplus 0)(0 \oplus 1)(1 \oplus 0)$

अर्थात् **(A) = 1 1 0 0 0 1 1 1**

➤ नोट–AND, OR, Exclusive-OR, Complement तथा SWAP A आदि प्रक्रियाओं में Flags प्रभावित नहीं होते हैं। लॉजिक निर्देशों के SET को तालिका 4.3 में प्रदर्शित किया गया है।

**तालिका 4.3 : Logical Instructions of MCS-51**

| Mnemonic | Description | Bytes | OSC Per |
|---|---|---|---|
| ANL A,Rn | AND Register to Accumulator | 1 | 12 |
| ANL A,direct | AND direct byte to Accumulator | 2 | 12 |
| ANL A,@Ri | AND indirect RAM to ACC | 1 | 12 |
| ANL A,#data | AND immediate data to ACC | 2 | 12 |
| ANL direct,A | AND ACC to direct byte | 2 | 12 |
| ANL direct,#data | AND immediate data to direct byte | 3 | 24 |
| ORL A,Rn | OR Register to ACC | 1 | 12 |
| ORL A,direct | OR direct byte to Accumulator | 2 | 12 |
| ORL A,@Ri | OR indirect RAM to ACC | 1 | 12 |
| ORL A,#data | OR immediate data to ACC | 2 | 12 |
| ORL direct,A | OR Accumulator to direct byte | 2 | 12 |
| ORL direct,#data | OR immediate data to direct byte | 3 | 24 |
| XRL A,Rn | Exc-OR Register to ACC | 1 | 12 |
| XRL A,direct | Exc-OR direct byte to ACC | 2 | 24 |
| XRL A,@Ri | Exc-OR indirect RAM to ACC | 1 | 12 |
| XRL A,#@data | Exc-OR immediate data to ACC | 2 | 12 |
| XRL direct,A | Exc-OR ACC to direct byte | 2 | 12 |
| XRL direct,#data | Exc-OR immediate data to direct byte | 3 | 24 |
| CLR A | Clear Accumulator | 1 | 12 |
| CPL A | Complement Accumulator | 1 | 12 |



| | | | |
|---|---|---|---|
| RL A | Rotate Accumulator Left | 1 | 12 |
| RLC A | Rotate ACC Left through Carry | 1 | 12 |
| RR A | Rotate ACC right | 1 | 12 |
| RRC A | Rotate ACC right through Carry | 1 | 12 |
| SWAP A | Swap nibbles within the ACC | 1 | 12 |

## बिट प्रचालित निर्देश (Bit Operating Instructions)

इन निर्देशों द्वारा कुछ-विशेष रजिस्टरों की एक बिट की स्टेट को परिवर्तित किया जा सकता है। जिन SFRs की बिटों के स्थान (Location) को Address किया जा सकता है।

Accumulator (ACC or A-Register), B-Register, Ports (All Four Ports : P0, P1, P2 and P3), Timer/Counter Control Register (TCON), Interrupt Priority Control Register (IP), Interrupt Enable Control Register (IE) and Programme Status World Register (PSW).

बिट प्रचालित निर्देश, एक ऑपरेण्ड (One Operand) अथवा दो ऑपरेण्ड (Two Operands) वाले होते हैं जिनमें गन्तव्य-बिट (Destination Bit) तथा स्रोत-बिट (Source Bit) ऑपरेण्ड होते हैं। इसके अतिरिक्त, किसी विशेष बिट की स्थिति (Status-High अर्थात् 'I' अवस्था अथवा Low अर्थात् '0' अवस्था) पर प्रतिबन्धित (Conditional) कुछ निर्देश होते हैं, जैसे—

(i) JC rel : Jump at the Addressed Location (Label) if Carry (C) in Set (*i.e.*, C = 1)

जहाँ, rel = Any Label

माना प्रोग्राम मैमोरी में लिखित प्रोग्राम में निर्देश अनुक्रम निम्न प्रकार से है—

```
          ..........................
          ..........................
          JC SUBR1
          ..........................
          ..........................
SUBR1 :   ..........................
          ..........................
          ..........................
```

CPU द्वारा इस प्रोग्राम के Execution के समय; जब Programme Counter (PC), JC SUBR1 निर्देश को Execute करेगा तो यदि उस समय Carry फ्लैग Set होगा (अर्थात् C = 1 होने पर) तो निर्देश से जो Label अथवा Address अंकित होता है, उस पर PC Jump करके पहुँच जाता है। इस उपरोक्त प्रोग्राम मे प्रयुक्त निर्देश में PC, लेबल SUBR1 पर पहुँच जाता है तथा CPU, यहाँ से अगले निर्देशों को अनुक्रम में Execute करना प्रारम्भ कर देता है परन्तु C = 1 न होने पर (अर्थात् C = 0 होने पर), CPU द्वारा JC SUBR1 निर्देश से अग्र निर्देशों का Execution अनुक्रम (Sequence) अनुसार होता रहता है।

**(ii) JCB Bit, rel :** Jump at the Addressed Location (Label) if a particular Bit of Given Register is set and also clear this Bit.

उदाहरणतया—यदि किसी प्रोग्राम में JBC निर्देश को इस प्रकार प्रयुक्त किया गया हो—

```
          ..........................
          ..........................
```

```
          JCB ACC.5 NEXT
          ........................
          ........................
NEXT :    ........................
          ........................
          ........................
```

तो CPU, इस प्रोग्राम के निर्देशों को निम्न प्रकार Execute करेगा—JCB ACC.5, NEXT निर्दे Execute करते समय, CPU निरीक्षण करेगा कि Accumulator की बिट नं० 5th, Set है कि नहीं (अर्थात् AC = High (1) अथवा Low (0) है)। यदि यह बिट Set होगी तो Programme Counter (PC), Jump करके NE लेबल पर पहुँच जाएगा तथा NEXT Location पर प्रयुक्त निर्देश तथा उसके अग्र निर्देशों को CPU द्वारा क्र अनुसार Execute किया जाता है तथा CPU द्वारा ACC.5 बिट को Clear अर्थात् ACC.5 = 0 कर दिया ज परन्तु ACC.5 बिट Set न हो (अर्थात् ACC.5 = 0 हो) तो पहले के समान, अनुक्रम अनुसार निर्देशों का CPU Execution होता रहेगा।

बिट प्रचालित निर्देशांकों को **बूलियन चल प्रचालन निर्देश** (Boolean Variable Manipulation Instructi भी कहते हैं। इस प्रकार के निर्देशों को तालिका 4.4 में प्रदर्शित किया गया है।

**4.4 : Boolean Variable Manipulation Instructions**

| Mnemonic | | Description | Bytes | OSC Peri |
|---|---|---|---|---|
| CLR | C | Clear Carry | 1 | 12 |
| CLR | bit | Clear direct bit | 2 | 12 |
| SETB | C | Set Carry | 1 | 12 |
| SETB | bit | Set direct bit | 2 | 12 |
| CPL | C | Complement Carry | 1 | 12 |
| CPL | bit | Complement direct bit | 2 | 12 |
| ANL | C, bit | AND direct bit to Carry | 2 | 24 |
| ANL | C,/bit | AND complement of direct bit to Carry | 2 | 24 |
| ORL | C,bit | OR direct bit to Carry | 2 | 24 |
| ORL | C,/bit | OR complement of direct bit to Carry | 2 | 24 |
| MOV | C,bit | Move direct bit to Carry | 2 | 12 |
| MOV | bit,C | Move Carry to direct bit | 2 | 24 |
| JC | rel | Jump if Carry is set | 2 | 24 |
| JNC | rel | Jump if Carry not set | 2 | 24 |
| JB | bit,rel | Jump if direct bit is Set | 3 | 24 |
| JNB | bit,rel | Jump if direct bit is not Set | 3 | 24 |
| JBC | bit,rel | Jump if direct bit is Set and Clear bit | 3 | 24 |

rd = Indicated Address or Label



## प्रोग्राम ब्रान्चिंग निर्देश (Programme Branching Instructions)

MCS-51 में प्रतिबन्धित (Conditional) और प्रतिबन्धन रहित (Unconditional) दोनों प्रकार के छलांग प्रचालन (Jump or Branch Operations) संभव है। ये Jump, Call एवं Return निर्देश हैं।

### (1) प्रतिबन्धित प्रोग्राम ब्रान्चिंग निर्देश (Conditional Programme Branching Instructions)

JZ, JNZ, CJNE तथा DJNZ आदि प्रतिबन्धित-प्रोग्राम ब्रान्चिंग निर्देश के स्मृति-सहायक (Mnemonics) हैं जिनका प्रतिबन्ध, 1-Byte word की स्थिति (status) पर आधारित होता है तथा यह एक बाइट, किसी दिये हुए SFR के contents अथवा किसी एक मैमारी Location में स्टोर Contents अथवा निर्देश में सीधा दिया हुआ हो सकता है। इनके प्रचालन का वर्णन तालिका 4.5 में किया गया है।

### (2) प्रतिबन्ध-रहित प्रोग्राम ब्रान्चिंग निर्देश (Unconditional Programme Branching Instructions)

प्रतिबन्ध-रहित निम्न स्मृति-सहायक (Mnemonics) निर्देश भी प्रोग्राम में प्रयोग किये जाते है—

(i) **SJAM rel :** Short Jump to the Level

माना प्रोग्राम मैमोरी में निम्न प्रोग्राम स्टोर हैं—

```
ORG 0105H      ;  Indicating Address of the Programme
.............  ;  Let an Instruction of 2-Byte
.............  ;  Let an Instruction of 1-Byte
SJAM READ      ;  Uncondition Jump to the "READ" Label
                  (2-Byte Instructions)
.............  ;  Let an Instruction of 3-Byte
MOVAA,57H      ;  Move the Contents of Internal RAM- Location 57 Hand and
                  its Instruction also labelled as "READ" (2-Byte Instruction)
.............  ;  Let an Instruction of 3-Byte
.............  ;  Let an Instruction of 1-Byte
SJAM READ      ;  Unconditional Jump to the "READ" Label (2-Byte Instruction)
...................
...................
```

CPU द्वारा ORG 0105H निर्देश Execute होते ही प्रोग्राम काउन्टर (PC) में निम्न Address Location स्टोर हो जाता है—

$$(PC) = 0105H$$

अत: प्रोग्राम अनुसार, दो निर्देशों के पश्चात् प्रथम SJAM READ निर्देश को Execute करते समय, प्रारम्भ में PC में स्टोर Location का मान

$$(PC) = 0105H + 02H + 01H = 0108H \text{ होता है}$$

अत: इस प्रथम SJAM READ निर्देश का CPU में Fetch होते ही (PC) के Contents में 02H की वृद्धि हो जाती है क्योंकि यह 2-Byte का निर्देश है अत: PC में स्टोर संख्या—

$$(PC) = 0108H + 02H = 010AH$$

तथा "READ" Label पर निर्देश (MOV, A, 57H), 3-Byte के पश्चात् है। अत: प्रथम SJAM READ निर्देश के कारण PC में निम्न संख्या स्टोर हो जाती है—

$$(PC) = 010\ AH + 03\ H = \mathbf{010DH}$$

(अर्थात यहाँ rel address = 03H)

अब 010DH Location से अग्र अनुक्रम (Sequence) अनुसार, CPU द्वारा निर्देश Execute किये जाते, जब PC, दूसरे SJAMP READ पर पहुँचता है तो PC में स्टोर संख्या—

$$(PC) = 010DH + 02H + 03H + 01H = 0113H$$

इस दूसरे निर्देश SJMP READ के CPU में Fetch होते ही, PC की संख्या निम्न हो जाती है—

$$(PC) = 0113H + 02H = 0115H$$

इसमें Relative Address (rel. Address) निम्न होगा—

$$\text{rel Address} = (2\ \text{Byte of SJAM}) + 01H + 03H + (2\text{-Byte of MOVE A, 57H}) = -08H$$

अत: दूसरे SJMP READ निर्देश के Execution के पश्चात् PC में स्टोर Address Location निम्न होगा—

$$(PC) = 0116H - \text{rel. Address} = 0115H - 08H = 010DH$$

अत: इस दूसरे SJMP READ के पश्चात् पुन: READ Label अर्थात् 010DH Memory Location से द्वारा निर्देशों का Execution प्रारम्भ किया जायेगा।

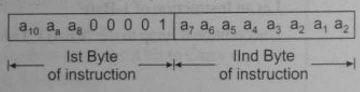
➤ **नोट–जिस memory Location पर SJAM rel निर्देश प्रयुक्त होता है उससे – 128 से + 127 Byte के मध्य ही rel Address का मान होना चाहिए।**

**(ii) LJMP addr-16 :** Long Jump to the given Address

इस निर्देश के Execute होने पर PC रजिस्टर में, निर्देश में प्रयुक्त 16-Bit Address संख्या स्टोर हो जाती और PC द्वारा इस संकेतिक Address से CPU, निर्देशों को Execute करना प्रारम्भ कर देता है। यह Address 64k-Byte प्रोग्राम में कहीं भी हो सकता है।

**(iii) AJMP addr. 11 :** Absolute Jump within 11-Bit Addresses (2k *i.e.*, 2048) from the Instruction following the AJMP Instruction.

11-Bits द्वारा अधिकतम 2k (अर्थात् 2048) Address Location को सम्बोधित कर सकते हैं। अत: मैमोरी में स्टोर प्रोग्राम में जिस Location पर निर्देश (AJMP) को प्रयुक्त किया जाता है उस Location से 2k ऊपर अथवा 2k नीचे प्रोग्राम के Execution को स्थानान्तरित (Transfer) किया जा सकता है।




$a_{10}$ ......... $a_0$ = 11-bits (LSBs) of 16-bit Address of Jmp Location is given in the Instruction

**चित्र 4.20 : Encode of AJMP Instruction**

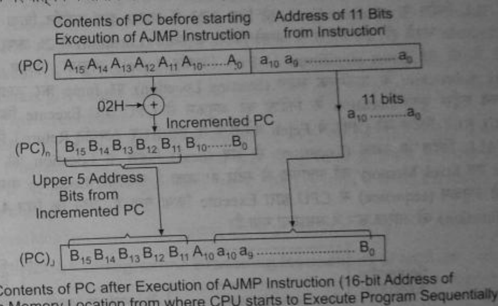
AJMP निर्देश के मशीन-कोड (Machine Code) अर्थात् Encoding को चित्र 4.20 से प्रदर्शित किया गया है। AJMP निर्देश Execution के पश्चात् प्रोग्राम के किस Address Location से Execution प्रक्रिया प्रारम्भ, अर्थात् Programme Counter (PC), प्रोग्राम के किस स्थान (Location) पर स्थानान्तरित हो जाए, इसकी गणना चित्र 4.21 में प्रदर्शित किया गया हैं, चूँकि AJMP निर्देश 2-Byte का है जैसा कि चित्र 4.20 में प्रदर्शित किया गया है। अत: CPU में इस निर्देश के Fetch होते ही (PC) में स्टोर 16-Bit Address Location ($A_{15}\ A_{14} \cdots A_0$) में 2 की वृद्धि होती है, अर्थात् $(PC)_n = (PC) + 2$ हो जाता है तथा चित्र 4.21 के अनुसार, माना PC में न्यू संचित (stored) Contents $(PC)_n = B_{15}B_{14} \ldots B_0$ है। इस निर्देश (AJMP) के Execution के पश्चात् प्रोग्राम नियंत्रण (Programme Control) जिस स्थान पर Jump करता है अर्थात् PC में जो प्रोग्राम स्थान (Location) स्टोर होता है उसके 16-Bits में ऊपर वाली 5-Bits तो $(PC)_n$ की ही होती है अर्थात् $B_{15}B_{14}B_{13}B_{12}B_{11}$ बिट तथा



11-Bits (11-LSB) निर्देश में दिये 11-बिट Address ($a_{10} \ldots a_0$) वाली रहती है। अत: AJMP निर्देश के Execution के पश्चात् PC में स्टोर contents अर्थात् प्रोग्राम स्थान $(PC)_5$ निम्न प्राप्त होता है—

$$(PC)_J = B_{15}\ B_{14}\ B_{13}\ B_{12}\ B_{11}\ a_{10}\ a_{10} \ldots a_0$$

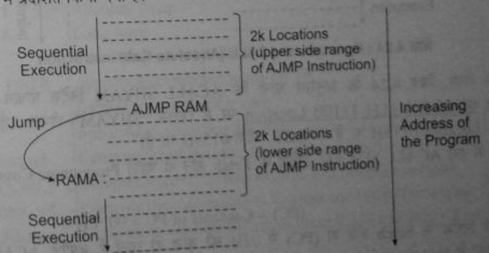
जिसके फलस्वरूप CPU, $(PC)_J$ प्रोग्राम स्थान (Location) से प्रोग्राम को अनुक्रमित (sequential) रूप से Execute करता है, जैसा कि चित्र 4.22 में प्रदर्शित किया गया है। इसमें माना कि AJMP RAMA निर्देश 07FEH Location पर है तथा लेबल (Label) RAMA, प्रोग्राम में 0857H = 00001 <u>00001010111B</u> Location पर है। अत: CPU में इस निर्देश के Fetch होने से पहले (PC) = 07FEH = 0000011111111110B है। इस निर्देश के CPU में Fetch होते ही $(PC)_n$ = 07FEH + 02H = 0800H = <u>00001</u> 00000000000B हो जाता है। अत: इस निर्देश के Execution के पश्चात् निम्न प्रोग्राम मैमोरी Address, PC में स्टोर, $(PC)_J$ हो जाता है—




**चित्र 4.21 : Destination Address Calculation in the Execution of AJMP and ACALL Instruction**

$$(PC)_J = \underline{00001}\ \underline{000}\ \underline{01010111B} = 0857H$$

इसके फलस्वरूप Programme control, 0857H Location अर्थात् RAMA लेबल पर Jump करता है तथा अब CPU द्वारा 0857H से, अनुक्रम (sequence) रूप में उच्च Addresses युक्त निर्देशों को Execute करता है, जैसा कि चित्र 4.22 में प्रदर्शित किया गया है।




**चित्र 4.22 : Function of AJMP (Absolute Jump) Instruction**

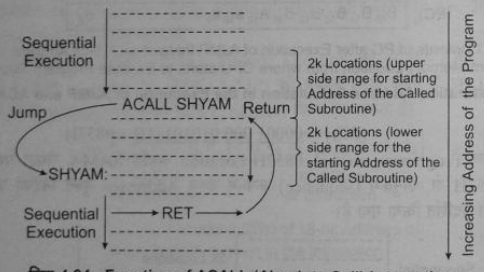
**(iv) ACALL Addr-11 :** Absolute Call the Subroutine within 11-Bit Address (2k *i.e.*, 2048) [of] the next Instruction following the CALL Instruction.

| $a_{10}$ $a_9$ $a_8$ 1 0 0 0 1 | $a_7$ $a_6$ $a_5$ $a_4$ $a_3$ $a_2$ $a_1$ $a_0$ |
|---|---|
| ← Ist Byte → | ← IInd Byte → |

4th - 0 bit of Ist Byte = 10001 = Machine code of ACALL Instruction
$a_{10}$ ......... $a_0$ = 11-Bits (LSBs) of 16-bit Address of the starting Location of Call Subroutine. These 11-bits of Address are given in the ACALL Instruction

**चित्र 4.23 : Encode of ACALL Instruction**

A CALL निर्देश के Machine Code को चित्र 4.23 में आरेख द्वारा प्रदर्शित किया गया है। CPU द्वारा निर्देश को Execute करते ही, इस निर्देश से अगले निर्देश के स्थान (Location) से 2k ऊपर अथवा नीचे Range मध्य निर्देश में दी गई Least significant 11-Bits (अर्थात् Label) अनुसार, प्रोग्राम नियन्त्रक (Program Controller), Subroutine के प्रारम्भिक स्थान (Starting Location) पर Jump कर जाता है। अब इस Jump स्थान से उच्च एड्रैस युक्त Location के निर्देशों को अनुक्रम से CPU द्वारा Execute किया जाता है। जब प्रोग्राम काउन्टर (PC), RET निर्देश को CPU में Fetch करता है तो इस RET (अर्थात् Return) निर्देश के Execution पश्चात् ACALL निर्देश के स्थान (Location) से अग्र (next) निर्देश के Location की Address, पुन: PC Stack Point एवं Stack Memory की सहायता से स्टोर हो जाता है तथा उस Location तथा उससे अग्र Location के निर्देशों को अनुक्रम (sequence) से CPU द्वारा Execute किया गया है, जैसा कि चित्र 4.24 में ACALL निर्देश की प्रक्रिया (function) को आरेख रूप में समझाया गया है।




**चित्र 4.24 : Function of ACALL (Absolute Call) Instruction**

उदाहरण के लिए, चित्र 4.24 के अनुसार माना कि ACALL SHYAM निर्देश प्रोग्राम मैमोरी में 00FE Location अर्थात् <u>00000 000 1111 1110B Location</u> पर है जिसमें "SHYAM" लेबल एड्रैस है तथा माना SHYAM लेबल, Location 011SH पर है एवं RET निर्देश 0159H पर है।

अत: CPU में इस ACALL SHYAM निर्देश के Fetch होने से पहले Programme Counter में स्टोर मैं एड्रैस (PC)—

$$(PC) = \text{Contents of PC} = 00FEH$$

CPU में इस निर्देश के Fetch होते ही (PC) में 02H की वृद्धि हो जाती है क्योंकि ACALL निर्देश 2-Byte का है, अत: अब PC में New Contents—

$$(PC)_n = 00FEH + 02H = 0100H$$



इस ACALL निर्देश के Execute होने पर, Stack-Pointer (SP) की सहायता से PC में स्थित ये Contents (प्रोग्राम में प्रयुक्त ACALL निर्देश से अग्र (next) निर्देश का Address), मैमोरी में एक निश्चित स्थान, जिसे Stack Memory कहते हैं, में स्टोर हो जाता है तथा निर्देश में दिया गया सांकेतिक Address (यहाँ पर SHYAM) Location की Address (माना 0115H है), PC में स्टोर हो जाती है, अत:

$$(PC)_{Call} = 0115H$$

**नोट**—ACALL निर्देश में Subroutine Programme के प्रारम्भिक Address की गणना AJMP निर्देश की भाँति ही चित्र 4.21 में प्रदर्शित आरेख के अनुसार ही करते है।

Subroutine के अन्त में RET अर्थात् Return निर्देश लिखा रहता है। जब CPU में RET निर्देश Execute होता है तो CPU, Stack-pointer की सहायता से Stack Memory में स्टोर Address 0100H (ACALL निर्देश से अग्र (next) निर्देश की (Location) को PC में स्थानान्तरित कर देता है। इसमे पुन: 0100H Memory Location से अनुक्रम रूप में, मुख्य प्रोग्राम (Main Program) Execute होने लगता है।

**(v) LCALL Addr-16 :** A Long Jump. Its Instruction allows jumping unconditionally in the programm at the address given inthe instruction. If instruction may be anywhere in the 64k external programme memory space (Location).

इस निर्देश की प्रक्रिया भी ACALL की भाँति ही होती है। LCALL एवं ACALL निर्देशों में अन्तर यह है कि LCALL निर्देश द्वारा Program में 64k Address के मध्य कहीं Jump किया जा सकता है जबकि ACALL निर्देश द्वारा जिस Location पर ACALL निर्देश प्रयुक्त किया गया है, उस Location से केवल 2K Range की Location पर ही Jump किया जा सकता है।

Programme Branching Instructions को तालिका 4.5 में प्रदर्शित किया गया है।

**तालिका 4.5 : Programme Branching Instructions of MCS-51**

| Mnemonic | | Description | Bytes | OSC Period |
|---|---|---|---|---|
| ACALL | addr 11 | Absolute Subroutine Call | 2 | 24 |
| LCALL | addr 16 | Long Subroutine Call | 3 | 24 |
| RET | | Return from Subroutine | 1 | 24 |
| RETI | | Return from Interrupt | 1 | 24 |
| AJMP | addr 11 | Absolute Jump | 2 | 24 |
| LJMP | addr 16 | Long Jump | 3 | 24 |
| SJMP | rel | Short Jump (relative address) | 2 | 24 |
| JMP | @A + DPTR | Jump indirect relative to the DPTR | 1 | 24 |
| JZ | rel | Jump if Accumulator is zero | 2 | 24 |
| JNZ | rel | Jum if Accumulator is not zero | 2 | 24 |
| CJNE | A,direct, rel | Compare direct byte to ACC and Jump if not equal | 3 | 24 |
| CJNE | A#data, rel | Compare immediate data to ACC and Jump if not equal | 3 | 24 |
| CJNE | Rn,#data, rel | Compare immediate data to Register and Jump if not equal | 3 | 24 |
| CJNE | @Ri,#data, rel | Compare immediate data to indirect RAM and Jump if not equal | 3 | 24 |
| DJNZ | Rn, rel | Decrement Register and Jump if not zero | 2 | 24 |
| DJNZ | direct, rel | No operation | 3 | 24 |

```
        JCB ACC.5 NEXT
        ........................
        ........................
NEXT :  ........................
        ........................
        ........................
```

तो CPU, इस प्रोग्राम के निर्देशों को निम्न प्रकार Execute करेगा—JCB ACC.5, NEXT निर्दे[illegible] Execute करते समय, CPU निरीक्षण करेगा कि Accumulator की बिट नं० 5th, Set है कि नहीं (अर्थात् A[illegible] = High (1) अथवा Low (0) है)। यदि यह बिट Set होगी तो Programme Counter (PC), Jump करके N[illegible] लेबल पर पहुँच जाएगा तथा NEXT Location पर प्रयुक्त निर्देश तथा उसके अग्र निर्देशों को CPU द्वारा [illegible] अनुसार Execute किया जाता है तथा CPU द्वारा ACC.5 बिट को Clear अर्थात् ACC.5 = 0 कर दिया [illegible] परन्तु ACC.5 बिट Set न हो (अर्थात् ACC.5 = 0 हो) तो पहले के समान, अनुक्रम अनुसार निर्देशों का CPU Execution होता रहेगा।

बिट प्रचालित निर्देशांकों को **बूलियन चल प्रचालन निर्देश** (Boolean Variable Manipulation Instruc[illegible] भी कहते हैं। इस प्रकार के निर्देशों को तालिका 4.4 में प्रदर्शित किया गया है।

**4.4 : Boolean Variable Manipulation Instructions**

| Mnemonic | | Description | Bytes | OSC Peri[illegible] |
|---|---|---|---|---|
| CLR | C | Clear Carry | 1 | 12 |
| CLR | bit | Clear direct bit | 2 | 12 |
| SETB | C | Set Carry | 1 | 12 |
| SETB | bit | Set direct bit | 2 | 12 |
| CPL | C | Complement Carry | 1 | 12 |
| CPL | bit | Complement direct bit | 2 | 12 |
| ANL | C, bit | AND direct bit to Carry | 2 | 24 |
| ANL | C,/bit | AND complement of direct bit to Carry | 2 | 24 |
| ORL | C,bit | OR direct bit to Carry | 2 | 24 |
| ORL | C,/bit | OR complement of direct bit to Carry | 2 | 24 |
| MOV | C,bit | Move direct bit to Carry | 2 | 12 |
| MOV | bit,C | Move Carry to direct bit | 2 | 24 |
| JC | rel | Jump if Carry is set | 2 | 24 |
| JNC | rel | Jump if Carry not set | 2 | 24 |
| JB | bit,rel | Jump if direct bit is Set | 3 | 24 |
| JNB | bit,rel | Jump if direct bit is not Set | 3 | 24 |
| JBC | bit,rel | Jump if direct bit is Set and Clear bit | 3 | 24 |

rd = Indicated Address or Label



## प्रोग्राम ब्रान्चिंग निर्देश (Programme Branching Instructions)

MCS-51 में प्रतिबन्धित (Conditional) और प्रतिबन्धन रहित (Unconditional) दोनों प्रकार के छलांग प्रचालन (Jump or Branch Operations) संभव है। ये Jump, Call एवं Return निर्देश हैं।

### (1) प्रतिबन्धित प्रोग्राम ब्रान्चिंग निर्देश (Conditional Programme Branching Instructions)

JZ, JNZ, CJNE तथा DJNZ आदि प्रतिबन्धित-प्रोग्राम ब्रान्चिंग निर्देश के स्मृति-सहायक (Mnemonics) हैं जिनका प्रतिबन्ध, 1-Byte word की स्थिति (status) पर आधारित होता है तथा यह एक बाइट, किसी दिये हुए SFR के contents अथवा किसी एक मैमारी Location में स्टोर Contents अथवा निर्देश में सीधा दिया हुआ हो सकता है। इनके प्रचालन का वर्णन तालिका 4.5 में किया गया है।

### (2) प्रतिबन्ध-रहित प्रोग्राम ब्रान्चिंग निर्देश (Unconditional Programme Branching Instructions)

प्रतिबन्ध-रहित निम्न स्मृति-सहायक (Mnemonics) निर्देश भी प्रोग्राम में प्रयोग किये जाते है—

**(i) SJAM rel :** Short Jump to the Level

माना प्रोग्राम मैमोरी में निम्न प्रोग्राम स्टोर हैं—

```
ORG 0105H      ;  Indicating Address of the Programme
..............  ;  Let an Instruction of 2-Byte
..............  ;  Let an Instruction of 1-Byte
SJAM READ      ;  Uncondition Jump to the "READ" Label
                   (2-Byte Instructions)
..............  ;  Let an Instruction of 3-Byte
MOVAA,57H      ;  Move the Contents of Internal RAM- Location 57 Hand and
                   its Instruction also labelled as "READ" (2-Byte Instruction)
..............  ;  Let an Instruction of 3-Byte
..............  ;  Let an Instruction of 1-Byte
SJAM READ      ;  Unconditional Jump to the "READ" Label (2-Byte Instruction)
....................
....................
```

CPU द्वारा ORG 0105H निर्देश Execute होते ही प्रोग्राम काउन्टर (PC) में निम्न Address Location स्टोर हो जाता है—

$$(PC) = 0105H$$

अतः प्रोग्राम अनुसार, दो निर्देशों के पश्चात् प्रथम SJAM READ निर्देश को Execute करते समय, प्रारम्भ में PC में स्टोर Location का मान

$$(PC) = 0105H + 02H + 01H = 0108H \text{ होता है}$$

अतः इस प्रथम SJAM READ निर्देश का CPU में Fetch होते ही (PC) के Contents में 02H की वृद्धि हो जाती है क्योंकि यह 2-Byte का निर्देश है अतः PC में स्टोर संख्या—

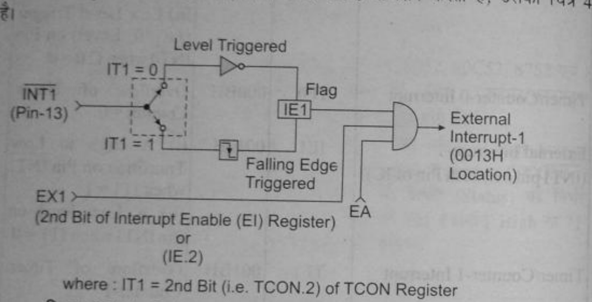
$$(PC) = 0108H + 02H = 010AH$$

तथा "READ" Label पर निर्देश (MOV, A, 57H), 3-Byte के पश्चात् है। अतः प्रथम SJAM READ निर्देश के कारण PC में निम्न संख्या स्टोर हो जाती है—

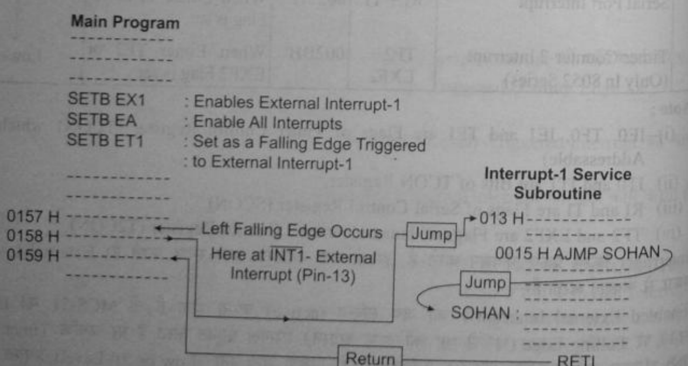
$$(PC) = 010\ AH + 03\ H = \mathbf{010DH}$$

मैमोरी में संचित (Store) हो जाते हैं तथा उस इन्ट्रप्ट (यहाँ External Interrupt-1) की Vector Address 0013 प्रोग्राम काउण्टर (PC) में लोड (Load) हो जाता है। जब किसी Generated Interrupt की Vector Address, प्रो गणक (PC) में लोड हो जाती है तो उस Interrupt का उत्पन्न Flag (यहाँ IE1) स्वत: Hardware द्वारा Clear ( IE = 0) हो जाता है परन्तु यदि External Interrupt-0 अथवा External Interrupt-1, Low Level Triggered Mode में हो तो उसे उसकी पिन (क्रमश: $\overline{INT0}$ अथवा $\overline{INT1}$) पर High अवस्था प्रयुक्त कर उसके Flag (क्रमश: IE0 अथवा IE1) को Clear किया जा सकता है। अब CPU, 0013H Location से "External Interrupt Service Subroutine ISS" के निर्देशों को अनुक्रम अनुसार Fetch तथा Execute करता रहता है जब तक कि इस ISS RETI निर्देश Execute नहीं होता है। ISS के अन्त में लिखे RETI निर्देश Execute करने पर CPU द्वारा, Stack मैमोरी में स्टोर मुख्य प्रोग्राम के पुराने Addresh, पुन: PC में Load हो जाते हैं। इसके फलस्वरूप CPU में, मुख्य प्रोग्राम के पुराने Address से ही पुन: निर्देश अनुक्रम अनुसार Fetch तथा Execute होते रहते हैं।

External Interrupt-1 को IC के अन्दर उत्पन्न करने के ब्लॉक-आरेख को चित्र 4.25 (a) में प्रदर्शित किया गया है, तथा External Interrupt-1 उत्पन्न होने पर, किस प्रकार CPU फ़ंक्शन करता है, उसको चित्र 4.25 (b) में प्रदर्शित किया गया है।




चित्र 4.25 (a) : An idea of Interrupt-1 Sources of MCS-51




चित्र 4.25 (b) : Function Diagram of Vector Interrupt-1 with an Example



चूँकि प्रत्येक Interrupt को उसकी Interrupt Service Subroutine (ISS) को स्टोर करने के लिए निश्चित Vector Address युक्त केवल 8 Locations (जैसे External Interrupt-1 को ISS के लिए 0013H उससे 001AH) स्थान दिये गये हैं जोकि प्राय: किसी Subroutine के लिए अप्राप्त स्थान है अत: इस Locations के अन्दर प्राय: Jump निर्देश का प्रयोग किया जाता है जिससे Programme Controller (PC) उस अंकित Jump स्थान पर स्थानान्तरित हो जाता है तथा इस सब-प्रोग्राम (ISS) के अन्त में RETI प्राप्त होने पर, Programme Controller पुन: Main Programme के उसी स्थान पर लौट जाता है जहाँ से उसे छोड़ा था, जैसा कि चित्र 4.25 (b) में प्रदर्शित किया गया है।

## MCS-51 के बाह्य इन्ट्रप्ट (External Interrupts of MCS-51)

जैसा कि पूर्व में वर्णन किया जा चुका है कि—MCS-51 श्रेणी के सभी माइक्रो-कन्ट्रोलरों में दो बाह्य इन्ट्रप्ट होते हैं—

**(i) बाह्य इन्ट्रप्ट-0 (External Interrupt-0)**—जिसके फंक्शन को, चिप की $\overline{INT0}$-पिन (पिन-12) पर उचित सिग्नल प्रयुक्त कर प्राप्त किया जा सकता है।

**(ii) बाह्य इन्ट्ररुप्त-1** (External Interrupt-1)—जिसके फंक्शन को, चिप की $\overline{INT1}$ पिन (पिन-13) परउचित सिग्नल प्रयुक्त कर प्राप्त किया जा सकता है।

दोनों बाह्य इन्ट्रप्टों की बाह्य स्रोतों (External Sources) को Software प्रोग्राम द्वारा निर्धारित किया जा सकता है कि इन्ट्रप्ट की बाहरी पिन पर शून्य स्तर विभव (Zero Level Voltage) होने पर अथवा High से Low ('1' से '0') स्तर के परिवर्तनीय समय में सिग्नल की (Falling Edge अर्थात् Transition-Time) पर बाह्य इन्ट्रप्ट सक्रिय (Active) हो। इसके लिए, Software प्रोग्राम द्वारा SRFs के TCON (Time Control) Register की IT0 अथवा/और IT1 बिट को उचित Set (अर्थात् '1') अथवा Clear (अर्थात् '0') अवस्था प्रयुक्त की जाती है। जब ITX = 0 अथवा ITX = 1 अवस्था में हो तो बाह्य इन्ट्रप्ट निम्न प्रकार से प्रक्रिया करते हैं—

ITX = 0 : External Interrupt-X एक स्तर-सक्रिय (Level Activated) इन्ट्रप्ट की भाँति कार्य करता है अर्थात् जब $\overline{INTX}$ पिन पर निम्न स्तर (Low Level अर्थात् '0' अवस्था) प्राप्त होती है तभी वह इन्ट्रप्ट ट्रिगर (Trigger) करता है।

ITX = 1 : External Interrupt-X (जहाँ, $X = 0$ अथवा 1 है) एक परिवर्तन-सक्रिय (Transition-Activated) इन्ट्रप्ट की भाँति कार्य करता है, अर्थात् जब $\overline{INTX}$, पिन पर प्राप्त सिग्नल की Falling Edge पर (High से Low की तरफ अग्रस्त अर्थात् '1' से '0' अवस्था की तरफ अग्रस्त Edge पर) इन्ट्रप्ट ट्रिगर करता है।

जहाँ, X = 0 (बाह्य इन्ट्रप्ट-0 के लिए) अथवा X = 1 (बाह्य इन्ट्रप्ट-1 के लिए) है।

चूँकि बाह्य इन्ट्रप्ट की पिनें ($\overline{INT0}$; पिन नं०-12 तथा $\overline{INT1}$; पिन नं०-13), की स्थिति (status) को प्रत्येक मशीन चक्र (Machine Cycle) के अन्त में परखा (Sample किया) जाता है तथा 1 Machine Cycle = 12 Oscillator Clocks (Oscillator Periods) होता है अत: बाह्य इन्ट्रप्ट पिन पर कम से कम एक मशीन चक्र अर्थात् 12 Oscillator Periods तक निम्न स्तर (Low Level) अथवा उच्च स्तर (High Level) होना चाहिए ताकि पिन की परख (Sampling) सुनिश्चित (ensure) हो जाए।

यदि बाह्य इन्ट्रप्ट परिवर्तनीय सक्रिय (Transition-Activated) अर्थात् Falling Edge Triggered अवस्था में स्थित हो तो कम से कम एक मशीन चक्र (अर्थात् 12 Oxcillator Periods) के लिए अनुरूप बाह्य पिन (Corresponding External-Pin; $\overline{INT0}$ पिन अथवा $\overline{INT1}$ पिन) पर प्रयुक्त स्रोत (Source) को निम्न स्तर (Low Level) पर अवश्य होना चाहिए जिससे यह सुनिश्चित हो सके कि पिवन पर विभव परिवर्तन (transition) को माइक्रो-कन्ट्रोलर द्वारा देख लिया गया है तथा जिसके फलस्वरूप इन्ट्रप्ट स्वीकार करने की प्रार्थना करने वाला Flag IEX (जहाँ X = 0 अथवा 1 है) सैट हो सके। Flags IE0 एवं IE1, टाइमर नियन्त्रक रजिस्टर (TCON) जोकि Bit Addressable है कि क्रमश: Ist Bit (TCON.1) एवं 3rd Bit (TCON.3) होती है। TCON रजिस्टर का चित्र सहित

वर्णन अग्र खण्ड में करेंगे। जब बाह्य इन्ट्रप्ट Edge-Trigger द्वारा सक्रिय हो अर्थात् IEX Flag उत्पन्न हो तो CP द्वारा IEX Flag स्वत: Clear अर्थात् Reset ('0' अवस्था युक्त) हो जाता है, जब CPU द्वारा एक इन्ट्रप्ट सर्वि सबरुटीन (Interrupt Service Subroutine; ISS) को Call करता (अर्थात् Execute करना प्रारम्भ) करता है।

यदि बाह्य इन्ट्रप्ट, स्तर-सक्रिय (Level-Activated) है तो अनुरूप पिन पर संयोजित बाह्य-स्रोत (Extern Source) को तब तक निम्न अवस्था (Low State) में रहना चाहिए जब तक कि इन्ट्रप्ट वास्तव में उत्पन्न हो जाता अर्थात् जब तक कि ISS को CPU द्वारा CALL नहीं कर लिया जाता है क्योंकि यदि Flag IEX के Set होने पर इन्ट्रप्ट इन्तजार (Wait) अवस्था में है तो बाहरी अनुरूप पिन से Low स्टेट हटाते ही Flag IEX क्लियर अर्थात् Res अवस्था में पहुँच जाता है जिससे CPU द्वारा इस इन्ट्रप्ट को नहीं स्वीकार किया जायेगा। जब एक बार CPU द्वारा बाह्य इन्ट्रप्ट को स्वीकार कर लिया जाता है अर्थात् उसकी ISS को Execute करना प्रारम्भ कर दिया जाता है तो इन्ट्रप्ट की बाहरी पिन ($\overline{INT0}$ अथवा $\overline{INT1}$ पिन) पर से निम्न अवस्था को हटाया जा सकता है अथवा एक और उस इन्ट्रप्ट को उत्पन्न किया जा सकता है।

## MCS-51 में टाइमर तथा काउन्टर (Timers and Counters in MCS-51)

MCS-51 श्रेणी के माइक्रो-कन्ट्रोलरों में चिप में ही Timer द्वारा एक निश्चित समय पश्चात् (Time Delay) कोई प्रक्रिया प्राप्त की जा सकती है अर्थात् समय की गणना की जा सकती है तथा Counter द्वारा इसके इनपुट प्रयुक्त स्पंदों (Pulses) की गणना की जा सकती है। Timer/Counter द्वारा आवृत्ति मापन (Fequency Measurement), स्पंद-चौड़ाई मापन (Pulse width measurement), स्पंद गणना (Pulse Counting) श्रेणी पोर्ट के लिए बॉड-दर उत्पन्न करना (Baud Rate Generation) इत्यादि प्रक्रियायें प्राप्त की जा सकती हैं।

MCS-51 श्रेणी की 8051-सब श्रेणी के माइक्रो-कन्ट्रोलरों (3051, 30C51, 8051, 80C51, 8751 एवं 87C51) में दो Timers/Counters होते हैं, जिनको Timer-0/Counter-0 एवं Timer-1/Counter-1 नामों से पुकारा जाता है जबकि 8052-सब-श्रेणी के माइक्रो-कन्ट्रोलरों (3052, 30C52, 8052, 80C52, 8752 एवं 87C52) में इन दो Timers/Counters (Timers-0/Counter-0 तथा Timer-1/Counter-1) के अतिरिक्त एक ओर Timer/Counter होता है जिसे Timer-2/Counter-2 नाम से पुकारते हैं। MCS-51 श्रेणी के सभी Timers/Counters, 16-Bits (=2 Bytes) के हैं। एक 16-बिट टाइमर की उच्चतम् बाईट (Higher Byte) को THX तथा Lower Byte को TLX द्वारा निर्दिष्ट (referred) करते हैं। जहाँ, X = 0, 1 या 2 है। उदाहरण के लिए टाइमर-1 की उच्चतम बाइट TH1 तथा निम्नतम बाइट TL1 है। एक टाइमर एवं एक काउन्टर में निम्न विभिन्नता होती है।

### टाइमर (Timer)

टाइमर, मशीन-चक्रों (Machine cycles) की गणना करता है तथा एक निश्चित समय विलम्ब (Reference Delay Time) अथवा वांछित चौड़ाई (Width) की High State युक्त वांछित आवृत्ति की क्लॉक प्रदान करता है। MCS-51 की एक मशीन चक्र में 12 दोलित्र समय-काल (12 Oscillator Periods अर्थात् 12 Oscillator Clocks) होते हैं अर्थात् Machine cycle की आवृत्ति ($f_{mc}$) का मान दोलित्र की आवृत्ति ($f_{0c}$) का 1/12 होता है। अत:

One Machine Cycle Period = 12 Oscillator Periods (Clocks)

Frequency of Machine Cycle ($f_{mc}$) = 1/12 Frequency of Oscillator ($f_{oc}$)

प्राय: MCS-51 माइक्रो-कन्ट्रोलरों में 12 MHz आवृत्ति दोलित्र प्रयोग किया जाता है अत: इसकी 12 clocks का समय-काल (12 Oscillator Periods अर्थात् One Machine Cycle का समय-काल (Period; $t_{mc}$) का मान $= 12/(12 \times 10^6)$ sec. $= 1 \times 10^{-6}$ sec $= 1\,\mu s$ होता है।

### काउण्टर (Counter)

MCS-51 के किसी एक काउण्टर की बाहरी इनपुट पिन (8051-श्रेणी में या तो T0 पिन अथवा T1 पिन तथा 8052-श्रेणी में या तो T0 या T1 या T2 पिन) पर सिग्नल की Falling Edge (Transition from '1' to '0') प्राप्त होने पर, काउण्टर के Contents में 1-अंक की वृद्धि (increment) हो जाती है। अत: एक काउण्टर, उसकी बाह्य पिन पर प्राप्त होने वाले '1' से '0' परिवर्तनों (Transitions) की अर्थात् सिग्नल की Falling Edges की गणना (Counting) को आउटपुट पर प्रदान करता है अथवा काउण्टर की इनपुट पिन पर सिग्नल की Falling Edge (from '1' to '0' transition) प्राप्त होने पर काउण्टर के आउटपुट पर संयोजित युक्ति पर अनुक्रिया (response) प्रदान करती है। काउण्टर की बाहरी पिन पर, Falling Edge (from '1' to '0' transition) का पता लगाने (detect) के लिए, काउण्टर 2-Machine Cycles अर्थात् 24 Oscillator Periods (= 2 μ sec) समय लेता है।



जब कोई टाइमर अथवा काउण्टर के Contents, FFFFH से 0000H पर ओवर-फ्लो (overflow) करते है तो यह ओवर-फ्लो, उस काउण्टर के Flag (Timer-0, के TF0 Flag को अथवा Timer-1 के TF1 Flag को) Set (अर्थात् 0 से 1 अथवा में) करता है तथा यह Flag इस टाइमर/काउण्टर के इन्ट्रप्ट (Interrupt) को उत्पन्न करता है। टाइमर-2 का ओवर-फ्लो TF2 Flag को Set करता है। TF0 एवं TF1 Flag, Timer/Counter Control Register (TCON) की क्रमश: 5th-Bit (TCON 5) एवं 7th Bit (TCON.7 अर्थात् MSB) होती है जबकि TF2 Flag, Timer/Counter-2 Control Register (T2CON रजिस्टर) की 7th-Bit (T2CON.7 अर्थात् MSB होती है।)

## श्रेणी पोर्ट संचारण (Serial Port Communication)

डिजिटल डाटा संचारण में प्राय: श्रेणी डाटा स्थानान्तरण (Transmission) का ही प्रयोग किया जाता है। सामानान्तर डाटा संचारण की अपेक्षा श्रेणी डाटा संचारण में सबसे प्रमुख लाभ यह है कि श्रेणी संचारण में कम तारों (wires) की आवश्यकता होती है।

MCS-51 के सभी माइक्रो-कन्ट्रोलर के श्रेणी-पोर्ट, एक पूर्ण द्विदिश पोर्ट (Full Duplex Serial Port) है। पूर्ण द्विदिश पोर्ट का अर्थ है कि एक ही समय पर 1 Byte (8-Bit) डाटा को Micro-controller से बाहर भेज (Transmit) सकते हैं तथा बाहर से अन्दर ग्रहण (Receive) भी कर सकते हैं। इसके अतिरिक्त, MCS-51 के श्रेणी पोर्ट, रिसीव बफर (Receive- Buffered) प्रकार के हैं अर्थात् जब CPU एक बाइट को पढ़ (Read) रहा हो तो उस समय एक अन्य Byte के प्राप्त होने पर श्रेणी-पोर्ट इस न्यू Byte को Buffer में स्टोर कर लेता है परन्तु इसी मध्य एक अन्य 3rd Byte भी प्राप्त हो जाये तो 2nd एवं 3rd Byte में से केवल एक Byte ही Buffer में स्टोर हो सकती है अत: दूसरी Byte खत्म (lost) हो जायेगी। MCS-51 श्रेणी-पोर्ट संचारण, RS-232 Standard के अनुसार प्रचालित होता है। इसमें RS निम्न सम्बन्ध को संकेत करता है—"Recommanded Standard Communication between two Micro-controllers/Microprocessors"।

MCS-51 के श्रेणी पोर्ट द्वारा माइक्रो-कन्ट्रोलर से डाटा बाहर भेजने (transmit) के लिए, बाहर से माइक्रो-कन्ट्रोलर में डाटा ग्रहण करने के लिए, दोनों स्थिति में CPU के Special Function Registers (SFRs) में से Serial Data Buffer (SBUF) रजिस्टर का प्रयोग होता है। अत: माइक्रो-कन्ट्रोलर से डाटा बाहर स्थानान्तरित (Transmission) के लिए, उस डाटा को पहले SBUF रजिस्टर में स्टोर करना होता है और फिर SBUF में स्टोर इस डाटा को बाहर संयोजित युक्ति में स्थानान्तरित किया जाता है। इसी प्रकार, बाहर से प्राप्त (Received) डाटा, सर्वप्रथम SBUF में रिसीव होता है उसके पश्चात्, इस प्राप्त डाटा को SBUF से माइक्रो-कन्ट्रोलर के अन्दर ही उचित स्थान पर स्थानान्तरित किया जाता है।

### बॉड रेट (Baud Rate)

श्रेणी-पोर्ट डाटा स्थानान्तरण में, बॉड-रेट एक बहुत ही महत्वपूर्ण गुणक (factor) है। श्रेणी पोर्ट से 8-Bit डाटा की एक-बिट भेजने में वास्तव में लगने वाले समय ($t_{1-Bit}$) के व्युत्क्रम (reciprocal) मान को बॉड रेट कहते है, अर्थात्

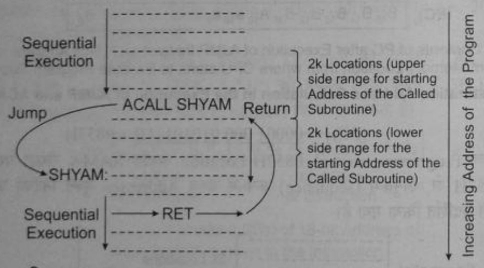
**(iv) ACALL Addr-11 :** Absolute Call the Subroutine within 11-Bit Address (2k *i.e.*, 2048) the next Instruction following the CALL Instruction.

| $a_{10}$ $a_9$ $a_8$ 1 0 0 0 1 | $a_7$ $a_6$ $a_5$ $a_4$ $a_3$ $a_2$ $a_1$ $a_0$ |
|---|---|
| ← Ist Byte → | ← IInd Byte → |

4th - 0 bit of Ist Byte = 10001 = Machine code of ACALL Instruction
$a_{10}$ ......... $a_0$ = 11-Bits (LSBs) of 16-bit Address of the starting Location of Call Subroutine. These 11-bits of Address are given in the ACALL Instruction

**चित्र 4.23 : Encode of ACALL Instruction**

A CALL निर्देश के Machine Code को चित्र 4.23 में आरेख द्वारा प्रदर्शित किया गया है। CPU द्वारा निर्देश को Execute करते ही, इस निर्देश से अगले निर्देश के स्थान (Location) से 2k ऊपर अथवा नीचे Range मध्य निर्देश में दी गई Least significant 11-Bits (अर्थात् Label) अनुसार, प्रोग्राम नियन्त्रक (Program Controller), Subroutine के प्रारम्भिक स्थान (Starting Location) पर Jump कर जाता है। अब इस Jump स्थान से उच्च एड्रैस युक्त Location के निर्देशों को अनुक्रम से CPU द्वारा Execute किया जाता है। जब प्र काउन्टर (PC), RET निर्देश को CPU में Fetch करता है तो इस RET (अर्थात् Return) निर्देश के Execution पश्चात् ACALL निर्देश के स्थान (Location) से अग्र (next) निर्देश के Location की Address, पुनः PC Stack Point एवं Stack Memory की सहायता से स्टोर हो जाता है तथा उस Location तथा उससे अग्र Location के निर्देशों को अनुक्रम (sequence) से CPU द्वारा Execute किया गया है, जैसा कि चित्र 4.24 में ACALL नि की प्रक्रिया (function) को आरेख रूप में समझाया गया है।




**चित्र 4.24 : Function of ACALL (Absolute Call) Instruction**

उदाहरण के लिए, चित्र 4.24 के अनुसार माना कि ACALL SHYAM निर्देश प्रोग्राम मैमोरी में 00FE Location अर्थात् <u>00000 000 1111 1110B Location</u> पर है जिसमें "SHYAM" लेबल एड्रैस है तथा माना SHYAM लेबल, Location 011SH पर है एवं RET निर्देश 0159H पर है।

अत: CPU में इस ACALL SHYAM निर्देश के Fetch होने से पहले Programme Counter में स्टोर मैं एड्रैस (PC)—

$$(PC) = \text{Contents of PC} = 00FEH$$

CPU में इस निर्देश के Fetch होते ही (PC) में 02H की वृद्धि हो जाती है क्योंकि ACALL निर्देश 2-Byte का है, अत: अब PC में New Contents—

$$(PC)_n = 00FEH + 02H = 0100H$$



इस ACALL निर्देश के Execute होने पर, Stack-Pointer (SP) की सहायता से PC में स्थित वे Contents (प्रोग्राम में प्रयुक्त ACALL निर्देश से अग्र (next) निर्देश का Address), मैमोरी में एक निश्चित स्थान, जिसे Stack Memory कहते हैं, में स्टोर हो जाता है तथा निर्देश में दिया गया सांकेतिक Address (यहाँ पर SHYAM) Location की Address (माना 0115H है), PC में स्टोर हो जाती है, अत:

$$(PC)_{Call} = 0115H$$

**नोट**—ACALL निर्देश में Subroutine Programme के प्रारम्भिक Address की गणना AJMP निर्देश की भाँति ही चित्र 4.21 में प्रदर्शित आरेख के अनुसार ही करते है।

Subroutine के अन्त में RET अर्थात् Return निर्देश लिखा रहता है। जब CPU में RET निर्देश Execute होता है तो CPU, Stack-pointer की सहायता से Stack Memory में स्टोर Address 0100H (ACALL निर्देश से अग्र (next) निर्देश की (Location) को PC में स्थानान्तरित कर देता है। इसमे पुन: 0100H Memory Location से अनुक्रम रूप में, मुख्य प्रोग्राम (Main Program) Execute होने लगता है।

**(v) LCALL Addr-16 :** A Long Jump. Its Instruction allows jumping unconditionally in the programm at the address given inthe instruction. If instruction may be anywhere in the 64k external programme memory space (Location).

इस निर्देश की प्रक्रिया भी ACALL की भाँति ही होती है। LCALL एवं ACALL निर्देशों में अन्तर यह है कि LCALL निर्देश द्वारा Program में 64k Address के मध्य कहीं Jump किया जा सकता है जबकि ACALL निर्देश द्वारा जिस Location पर ACALL निर्देश प्रयुक्त किया गया है, उस Location से केवल 2K Range की Location पर ही Jump किया जा सकता है।

Programme Branching Instructions को तालिका 4.5 में प्रदर्शित किया गया है।

**तालिका 4.5 : Programme Branching Instructions of MCS-51**

| Mnemonic | | Description | Bytes | OSC Period |
|---|---|---|---|---|
| ACALL | addr 11 | Absolute Subroutine Call | 2 | 24 |
| LCALL | addr 16 | Long Subroutine Call | 3 | 24 |
| RET | | Return from Subroutine | 1 | 24 |
| RETI | | Return from Interrupt | 1 | 24 |
| AJMP | addr 11 | Absolute Jump | 2 | 24 |
| LJMP | addr 16 | Long Jump | 3 | 24 |
| SJMP | rel | Short Jump (relative address) | 2 | 24 |
| JMP | @A + DPTR | Jump indirect relative to the DPTR | 1 | 24 |
| JZ | rel | Jump if Accumulator is zero | 2 | 24 |
| JNZ | rel | Jum if Accumulator is not zero | 2 | 24 |
| CJNE | A,direct, rel | Compare direct byte to ACC and Jump if not equal | 3 | 24 |
| CJNE | A#data, rel | Compare immediate data to ACC and Jump if not equal | 3 | 24 |
| CJNE | Rn,#data, rel | Compare immediate data to Register and Jump if not equal | 3 | 24 |
| CJNE | @Ri,#data, rel | Compare immediate data to indirect RAM and Jump if not equal | 3 | 24 |
| DJNZ | Rn, rel | Decrement Register and Jump if not zero | 2 | 24 |
| DJNZ | direct, rel | No operation | 3 | 24 |

5

# एसेम्बली लैंग्वेज प्रोग्रामि
## (Assembly Language Programm

## असेम्बली भाषा–लाभ तथा हानियाँ (Assembly Language : Advantages and Disadvantages)

### असेम्बली भाषा (Assembly Language)

असेम्बली भाषा, मशीन भाषा में आ रही परेशानियों को दूर करने के लिए तैयार की गई। इसमें मशीन क स्थान पर नेमोनिक कोड (mnemonic code) का प्रयोग किया गया, जिन्हें मानव मस्तिष्क आसानी से पहचान था। उदाहरण के लिए SUB (SUBTRACT), JMP/(JUMP) तथा इसी तरह के कई अन्य नेमोनिक कोड आसानी से पहचाना व याद रखा जा सकता था। प्रोग्राम इन्हीं संकेतों में लिखा जाने लगा। इनमें से प्रत्येक संकेत के एक मशीन कोड भी निर्धारित किया गया पर असेम्बली कोड से मशीन कोड में परिवर्तन का काम, कम्प्यूटर में ही एक प्रोग्राम के द्वारा किया जाने लगा। इस प्रकार के प्रोग्राम को असेम्बलर नाम दिया गया। मशीन और असेंबली चूँकि कम्प्यूटर की मूल संरचना से संबंद्ध हैं, एवं अलग-अलग संरचनाओं वाले कम्प्यूटर के लिए समान नहीं हो अत: इन्हें निम्नस्तरीय भाषाएँ कहा गया।

**असेम्बली भाषा के लाभ (Advantages of Assembly Language)**

असेम्बली भाषा के निम्नलिखित लाभ हैं—

(i) यह प्रोग्रामर के समय की बचत करता है।
(ii) निर्देशों में त्रुटियों को कम करता है।
(iii) त्रुटियाँ शीघ्र जाँची जा सकती हैं।
(iv) मशीनी भाषा की अपेक्षा असेम्बली भाषा के निर्देशों में बदलाव या नवीनीकरण शीघ्रता से किया सकता है।
(v) प्रोग्रामर को डाटा के मैमोरी में संग्रह स्थान की संख्या याद रखने की आवश्यकता नहीं पड़ती है।

**असेम्बली भाषा की हानियाँ (Disadvantages of Assembly Language)**

असेम्बली भाषा की निम्नलिखित हानियाँ हैं—

(i) यद्यपि यह भाषा मशीन भाषा से सरल है लेकिन प्रोग्रामिंग में अधिक समय लगता है और ये समझने में सरल भी नहीं होते हैं।
(ii) असेम्बली भाषा में तैयार प्रोग्राम में त्रुटियाँ ढूँढ़ना कठिन होता है।
(iii) असेम्बली भाषा भी मशीन भाषा पर आधारित है, अत: प्रत्येक कम्प्यूटर की स्वयं की एक असेम्बली भाषा होती है जिससे प्रत्येक कम्प्यूटर के लिए अलग-अलग प्रोग्राम तैयार करने पड़ते हैं।



## एसेम्बलर दिशासूचक (Assembler Directive)

MCS-51 फैमली के माइक्रो-कन्ट्रोलर की एसेम्बली लैंग्वेज में कुछ विशेष स्मृति-सहायक निर्देश (Mnemonic Instructions) होते हैं जोकि एसेम्बलर (Assembler) के लिए दिशासूचक का कार्य करते हैं इसलिए इन्हें एसेम्बलर दिशासूचक (Assembler Directives) कहते हैं। MCS-51 फैमली की एसेम्बली लैंग्वेज के प्रमुख एसेम्बलर दिशासूचक निर्देश निम्न हैं—

(i) ORG (Origin)
(ii) END (End of the main program)
(iii) DB (Define Byte)
(iv) EQU (Equate)

### (1) ORG, एसेम्बलर दिशासूचक (ORG : Origin Mnemonic of Assember Directives)

एसेम्बली लैंग्वेज का स्मृति-सहायक निर्देश—"ORG", MCS-51 के एसेम्बलर (जिसे 8051 का एसेम्बलर भी कहते हैं) के लिए एक आभासी-निर्देश (Pseudo-instruction) है जो कि ऐसम्बलर को सोर्स-प्रोग्राम (Source Programme अर्थात् Main Programme) अथवा कोई सब-प्रोग्राम (Sub-programme अर्थात् Sub-routine) के प्रारम्भिक एड्रैस (Beginning या Origin Address) को सूचित (अर्थात् आभास) करता है। उदाहरण के लिए—

ORG 0500H

निर्देश, एसेम्बलर को बताता है कि—प्रोग्राम मैमोरी में 0500H एड्रैस से सोर्स-प्रोग्राम प्रारम्भ (Start) है। उपरोक्त उदाहरण में, एड्रैस लोकेशन को हैक्सा-डेसीमल (Hexa-decimal as 0500H) में प्रदर्शित किया गया है। सोर्स-प्रोग्राम के इसी प्रारम्भिक एड्रैस को डेसीमल में भी निम्न प्रकार से प्रदर्शित कर सकते हैं—

ORG 1280D

### (2) END, एसेम्बली दिशासूचक (END : End Mnemonic of Assembler Directives)

एसेम्बली लैंग्वेज का स्मृति-सहायक निर्देश "END" भी MCS-51 (अथवा 8051) के एसेम्बलर के लिए एक आभासी निर्देश है जोकि एसेम्बलर को सोर्स-प्रोग्राम के समाप्त होने की सूचना प्रदान करता है अर्थात् एसेम्बली लैंग्वेज के लिखे सोर्स-प्रोग्राम की अन्तिम लाइन (Last-line) में END निर्देश लिखा रहता है।

MCS-51 फैमली के माइक्रो-कन्ट्रोलर द्वारा END निर्देश को Execute करने के पश्चात्, प्रोग्राम के निर्देशों के Execution की प्रक्रिया बन्द कर देता है। अत: यदि END निर्देश के पश्चात् यदि सोर्स-प्रोग्राम में कोई अन्य निर्देश लिखे हुए भी हों तो उन्हें CPU द्वारा नजर-अन्दाज (ignore) कर दिया जाता है अर्थात् न तो वे निर्देश CPU में Fetch होते हैं जिसके फलस्वरूप न ही Execute हो पाते हैं।

### (3) EQU, एसेम्बली दिशासूचक (EQU : Equate Mnemonic of Assembler Directives)

MCS-51 (अर्थात् 8051) की एसेम्बली-लैंग्वेज के "EQU" दिशासूचक निर्देश का उपयोग किसी एक स्थिरांक (Constant) को परिभाषित करने के लिए किया जाता है। इस निर्देश द्वारा किसी एक स्थिरांक (अर्थात् एक नम्बर) को किसी वांछित डाटा लेबल (Data Label—कोई अर्थ युक्त शब्द अथवा अर्थ-हीन, दो या दो से अधिक अक्षरों का शब्द) के साथ सम्बन्धित (associate) किया जाता है। इसके पश्चात् जब भी इस लेबल को प्रोग्राम में डाटा के स्थान पर प्रयोग किया जाएगा तो CPU द्वारा स्वत: ही Label के स्थान पर उस स्थिरांक (नम्बर) को Data में ले लेगा। उदाहरण के लिए—माना एक लेबल COUNT को 25H मान द्वारा नियत करना है तथा 25H नम्बर का वर्गमूल (Square Value) लेबल को निर्देश में प्रयोग करके ज्ञात करना है तो इसके लिए निम्न निर्देश प्रोग्राम में लिखते हैं—

```
COUNT EQU 25H  ;    COUNT = 25H
MOV A,#COUNT   ;    COPY A = COUNT = 25H
MOV B,#COUNT   ;    Copy B = COUNT = 25H
MUL AB         ;    Multiply A and B Store Result in
```

```
                  ;   BA (Higher Byte of Result in B
                  ;   Register and Lower Byte of
                  ;   Result in A-Register)
MOV 51H, A        ;   Store Lower Byte of Result
                  ;   in Data RAM in Location's
                  ;   Address—51H
MOV 52H, B        ;   Store Higher-Byte of Result
                  ;   in Data RAM at Location's
                  ;   Address-52H
```

इस प्रोग्राम द्वारा प्राप्त 25H × 25H = 0559H Result की Higher Byte अर्थात् यहाँ 05H तो SFRs के स्टोर होती है जबकि Lower Byte अर्थात् यहाँ 59H का स्टोर A रजिस्टर में होता है। प्रोग्राम के अन्त में दि निर्देश—MOV 51H, A तथा MOV 52H, B द्वारा परिणाम (0559H) की निम्न बाइट (59H) एवं उच्च (05H) का स्टोर क्रमश: डाटा RAM के एड्रेस 51H एवं 52H में हो जाता है।

इस प्रोग्राम में "COUNT EQU 25H" निर्देश द्वारा, COUNT लेबल का मान 25 हैक्सा-नम्बर के बरा जाता है। परन्तु 25H = 37 डेसी नम्बर होता है अत: यह निर्देश को इस प्रकार भी लिख सकते हैं—

COUNT EQU 37D

अथवा

COUNT EQU 37

इस दिशासूचक निर्देश "EQU" का प्रमुख लाभ यह है कि जब लेबल का प्रयोग प्रोग्राम में कई स्थान पर हो तो लेबल का मान, निर्देश के अनुसार परिवर्तित होता रहता है। इससे किसी विशेष घटना की आवृत्ति की गणना स से की जा सकती है तथा प्रोग्राम एक्सक्यूजन (Execution) की JUMP प्रक्रिया में लेबल अति सहायक होता है कि समय की काफी बचत होती है

**नोट**—एसेम्बली-लैंग्वेज प्रोग्राम को लिखते समय, स्मृति-सहायक (Mnemonics) निर्देशों (जैसे—MO ADD, SUB, INC, MUL इत्यादि) एवं एसेम्बलर दिशासूचक शब्दों (ORG, END, DB एवं EQU इत्यादि) व कुछ अन्य आरक्षित शब्द (Reserved Words—जैसे SFR, SCON, ACC, A, B, TCON, TR1, TR0 इत्यादि) प्रयोग लेबल (Label) के चयन करने के लिए नहीं किया जाता है। इसके अतिरिक्त, एक प्रक्रिया के लिए केवल लेबल (Label) को प्रोग्राम में उपयोग करना चाहिए।

### ■ (4) DB, एसेम्बली दिशासूचक (DB : Define Byte of Assembler Directives)

डिफाइन बाइट (DB), निर्देश MCS-51 फैमली के एसेम्बलर के लिए दिशासूचक (Directive) की भाँति का करता है। प्रोग्राम में "DB" निर्देश का प्रयोग एक-बाइट (8-Bit Wide) डाटा नम्बर को डिफाइन करने के लिए कि जाता है कि प्रयुक्त डाटा डेसीमल (Decimal), बाइनरी (Binary), हैक्सा-डेसीमल (Hexa-decimal) अथवा ASC नम्बर में है। DB निर्देश में नम्बर के पश्चात् जो अक्षर (Letter) प्रयुक्त किया जाता है, वह CPU को पहचान करा कि प्रयुक्त नम्बर किसी नम्बर पद्धति (Number System) का है। उदाहरण के लिए—

(i) हैक्सा-डेसीमल नम्बर में हैक्सा-डेसीमल में लिखी संख्या के पश्चात् "H"-अक्षर का प्रयोग किया जा जैसे—21H, 57H, 85H इत्यादि। उदाहरणतया

Data-Hex : DB 47H; Assigning Data in Hexa-decimal Form

(ii) डेसीमल नम्बर में डेसीमल में लिखी संख्या के पश्चात् "D"-अक्षर का प्रयोग किया जाता है परन्तु यदि सं के पश्चात् कोई भी अक्षर न प्रयोग किया जाए तो CPU द्वारा उस नम्बर को डेसीमल में ही ग्रहण करता है।



उदाहरणतया—

Data-Dec : DB 59D; Assigning Data in Decimal

अथवा

Data-Dec : DB 59 ; Assigning Data in Decimal

(iii) बाइनरी नम्बर में, बाइनरी में लिखी संख्या के पश्चात् Alphabet के "B" अक्षर का प्रयोग किया जाता है। उदाहरणतया—

Data-Bin : DB 10101001B; Assigning Data in Binary

(iv) प्रोग्राम में प्रयुक्त संख्या को यदि ASCII Characters में प्रयुक्त करना हो तो संख्या को एक अथवा दो कोटेशन-चिन्ह (Quotation-mark(s)) में प्रयुक्त किया जाता है। उदाहरणतया—

DATA-ASCII : DB '47' ; Assigning Data in ASCII-Code

अथवा

DATA-ASCII : DB "47" ; Assigning Data in ASCII-Code

यदि किसी शब्दों की माला (String) को प्रोग्राम में परिभाषित (Define) करना हो तो उसे निम्न प्रकार से करते है—

DEVELOP-5 : DB "Development Programme of Kalyanpur, Kanpur City"

DATA-ASCII : DB "7157"

DB दिशासूचक निर्देश (Directive Instruction) के पश्चात् कोटेशन चिन्हों के मध्य प्रयुक्त शब्दों की माला (String) अथवा/और संख्याओं (Number) को MCS-51 फैमली (अर्थात् 8051) का एसेम्बलर स्वत: ही इसके अनुरूप ASCII-Code में नियुक्त कर देता है।

## MCS-51 के एसेम्बलर की प्रक्रिया (Assembler Operation of MCS-51 or 8051)

MCS-51 का एसेम्बलर एक सॉफ्टवेयर (Software) है जोकि एसेम्बली लैंग्वेज सोर्स कोडों अर्थात् स्मृति-सहायक निर्देशों (Mnemonics) को मशीन कोड (Machine Code) में परिवर्तित करता है। उदाहरण के लिए—

(i) **MOV A, $R_n$** निर्देश (जोकि एसेम्बली लैंग्वेज में है) को एसेम्बलर निम्न मशीन-कोड में परिवर्तित करता है—

$11101r_2r_1r_0$ Machine-Code

जहाँ, $r_2r_1r_0 = R_n$ तथा $n = 0, 1, \ldots$ या 7

अत: MOV A, $R_5$ निर्देश को एसेम्बलर निम्न मशीन कोड में परिवर्तन करता है—

11101101 Machine Code

जहाँ, $r_2r_1r_0 = 101 = R_5$ है।

(ii) **MOV A, Direct** निर्देश को एसेम्बलर निम्न मशीन कोड में परिवर्तित करता है—

| 11100101 | Direct Address |
|---|---|
| ⟵ One Byte ⟶ | ⟵ One Byte ⟶ |

जहाँ, Direct Address का अर्थ है CPU के किसी Register का एड्रेस जोकि आन्तरिक RAM में 00H से FFH तक हो सकता है अर्थात् किसी एक SFR का एड्रेस है।

```
अत: MOV, A, F5H ;  Move the Contents of Internal RAM
                ;  Register of Location's Address F5H
                ;  in the Accumulator or (F5H)→(A)
```

निर्देश का मशीन-कोड निम्न होता है—

11100101 11110101

**(iii) ADD A,#data** अर्थात् (A) + # data → (A)

इस निर्देश को एसेम्बलर निम्न मशीन-कोड में परिवर्तित करता है—

| | |
|---|---|
| 00010100 | Immediate Data |
| ← One Byte → | ← One Byte → |

अत: ADD A, #27H [अर्थात् (A) + 27H → (A)] निर्देश को एसेम्बलर निम्न मशीन-कोड में परिवर्तित करता है—

0010 0100 0010 0111

**(iv) ADD, A@Ri** ; $(A) + (R_i) \rightarrow (A)$ and i = 0 or 1
; Add the Contents of ACC with
; the Contents of Internal Data RAM
Location addressed Indirectly through Register $R_i$ and
; Result is stored in the ACC

इस निर्देश को एसेम्बलर निम्न मशीन कोड में परिवर्तित करता है—

0010011i जहाँ, i = 0 अथवा 1

अत: एसेम्बलर ADD A,@ R0 निर्देश को 0010 0110 मशीन-कोड में तथा ADD A, @R1 निर्देश को 0010 0111 मशीन-कोड में परिवर्तित करता है।

**(v) CLR C ;** Clear Carry Bit अर्थात्
; 0 ⇒ (C)

इस निर्देश को एसेम्बलर निम्न मशीन-कोड में परिवर्तित करता है—

**1100 0011**

इसी प्रकार सभी निर्देशों को एसेम्बलर, एक बाइट अथवा दो बाइट के विभिन्न मशीन-कोडों में परिवर्तित करता है।

अत: MCS-51 फैमली के किसी माइक्रो-कन्ट्रोलर द्वारा किसी समस्या (Problem) का समाधान (या उत्तर) प्राप्त करने के लिए, निम्न प्रक्रियायें करनी पड़ती हैं—

(1) वांछित समस्या के अनुसारव शुद्ध (Correct), सरल एवं सभी आवश्यक आवश्यकताओं को पूर्ण करने वाला छोटा प्रोग्राम, एसेम्बली लैंग्वेज में लिखते हैं। इस प्रोग्राम को सोर्स फाइल (Source File) कहते हैं जिसको वांछित नाम से देने के पश्चात् उसका विस्तार .asm द्वारा किया जाता है। उदाहरण के लिए—एक एसेम्बली लैंग्वेज प्रोग्राम की सोर्स फाइल को माना "Project-5" नाम देना चाहते हैं तो इस फाइल नाम के पश्चात् .asm लगाकर नामकरण करते हैं अर्थात् सोर्स फाइल का सम्पूर्ण नाम PROJECT-5.asm रखते हैं।

एसेम्बली लैंग्वेज में प्रोग्राम को लिखने के लिए "**एडीटर** (EDITOR)" अथवा अन्वय वर्ड प्रोसेसर (Word Processor) का प्रयोग करते हैं जोकि ASCII-Code में फाइल को निर्मित करता है।

(2) यह सोर्स फाइल (उदाहरण अनुसार : "PROJECT-5.asm" सोर्स फाइल), एसेम्बलर (Assembler) के लिए इनपुट फाइल होती है जिसके प्रत्येक निर्देश (Opcodes) को एसेम्बलर, मशीन कोड (Machine Code) में परिवर्तित करके .Obj विस्तार युक्त ऑबजेक्ट फाइल (Object File) प्रदान करता है। अत: उदाहरण अनुसार PROJECT-5.Obj फाइल प्राप्त होती है। इसके अतिरिक्त एसेम्बलर के आउटपुट पर एक .1st विस्तार फाइल प्राप्त होती है। उदाहरण के अनुसार,

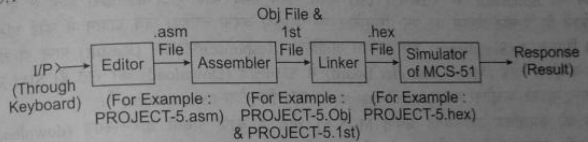
PROJECT-5.1st नाम युक्त फाइल भी एसेम्बलर के आउटपुट पर प्राप्त होती है। इस ... 1st फाइल में, सोर्स प्रोग्राम (....asm) में प्रयुक्त सभी Opcodes, Address तथा प्रोग्राम की एसेम्बली प्रक्रिया (Assembly Process) के मध्य प्राप्त त्रुटियों की पूर्ण सूची (list) होती है।



यदि आवश्यक हो तो इस फाइल को कुछ प्रकार के Editors में खोला (Open) जा सकता है। एक 8051 के लिए लिखे गये Assembly Language Programme को X-8051 पर एसेम्बल करने के पश्चात् X-8051.exe द्वारा प्रोग्राम को निष्पादित (Execute) कराया जा सकता है। एसेम्बलरों का लिनक्स अनुवाद (Linux Version of Assemblers) भी उपलब्ध है। उदाहरणतया—AS-31, एसेम्बलर का एक Linux Version है।

(3) अब एसेम्बलवर से प्राप्त Object Files को लिंकिंग (Linking) करने की आवश्यकता होती है। इसके लिए, Object File (उदाहरण अनुसार PROJECT-5.Obj फाइल) को Linker के इनपुट पर प्रदान करते हैं। लिन्कर सॉफ्टवेयर (Linker Software) जोकि प्राय: "Link.exe" नाम से होता है, इस प्रक्रिया को निष्पादित (execute) करता है। Linker द्वारा .hex विस्तार युक्त एक हैक्स फाइल (Hex File) निर्मित की जाती है। अत: उदाहरण अनुसार, Linker द्वारा निर्मित फाइल PROJECT-5.hex प्राप्त होती है। इसके अतिरिक्त, एक .bin फाइल भी प्राप्त हो सकती है जबकि HEX2BIN उपयोगिता का प्रयोग किया गया हो। MCS-51 के अनुरूपक (Simulator) इस .bin फार्मेट (उदाहरण अनुसार, PROJECT-5.bin) को स्वीकार कर सकता है।

MCS-51 के Editor, Assembler एव Linker की प्रक्रियाओं अर्थात् उनके द्वारा निर्मित फाइलों को उदाहरण सहित चित्र 5.1 में प्रदर्शित किया गया है।




**चित्र 5.1 : Operations of an Editor, an Assembler and a Linker of MCS-51 (or 8051) Family in Micro-controller**

## एसेम्बलर के लाभ (Advantages of the Assembler)

कम्प्यूटर/माइक्रो-कन्ट्रोलर की सहायता से उद्देश्य की पूर्ति के लिए प्रोग्राम को अभिकल्पित (design) करने तथा विकसित (develop) करने के लिए, एसेम्बलर एक अच्छा औजार (Tool) है। औद्योगिक आधार युक्त सॉफ्टवेयर लिखने के लिए एसेम्बलर एक अति आवश्यक टूल है क्योंकि मशीन भाषा (Machine Language) में किसी प्रोग्राम को लिखना/समझना बहुत अधिक कठिन कार्य है। स्मृति-सहायक (Mnemonics) निर्देशों को मशीन लैंग्वेज (Assembly Language को मशीन लैंग्वेज में परिवर्तित करना आवश्यक है क्योंकि कम्प्यूटर/माइक्रो-कन्ट्रोलर केवल मशीन लैंग्वेज को ही समझ सकता है) में अनुवाद करने के अतिरिक्त, अन्य कई प्रक्रियायें (functions) भी एसेम्बलर करता है। एसेम्बलर के कुछ प्रमुख लाभ निम्न हैं—

(i) एसेम्बलर, एसेम्बली लैंग्वेज में लिखे गये स्मृति-सहायक (Mnemonics) निर्देशों को मशीन कोड अर्थात् बाइनरी-कोड (Binary-Code) में अति शुद्धता तथा उच्च गति के साथ परिवर्तित करता है।
(ii) प्रोग्राम में प्रयुक्त किये गये संकेतों (symbols) को एसेम्बलर, उपयुक्त मान प्रदान करता है। एसेम्बलर का यह गुण, जम्प (Jump) निर्देश को सरल तथा प्रभावशाली बनाता है।
(iii) एसेम्बलर, वाक्य-रचना में हुई त्रुटियों (Syntax Errors) की खोज करता है, जैसे—गलत लेबलों अथवा/और व्यंजकों (expressions) का प्रयोग इत्यादि तथा त्रुटि-संदेशों को प्रादन करता है।
(iv) एसेम्बलर, प्रोग्राम में होने वाली लॉजिक-त्रुटियों की भी खोज करके, त्रुटि-संदेशों को प्रदान करता है।
(v) एसेम्बलर के कारण ही प्रोग्राम में सरलता से निर्देशों (Instructions) को सम्मिलित (insert) किया जाता है अथवा निकाल (delete) दिया जाता है।
(vi) एसेम्बलर, डाटा अथवा/और परिणाम (result) के लिए, मैमोरी-लोकेशन को सुरक्षित रख सकता है।

(vii) एसेम्बलर, प्रलेखन (documentation) के लिए, फाइलें प्रदान करता है।

(viii) एक एसेम्बली लैंग्वेज प्रोग्राम के परीक्षण (Test) एवं डिबग (Debug) में एसेम्बलर का प्रयोग ... जाता है।

## MCS-51 का सॉफ्टवेयर अनुरूपक (Software Simulator of MCS-51 or 8051)

MCS-51 फैमली के माइक्रो-कन्ट्रोलर-8051 का सॉफ्टवेयर सिमुलेटर, बाजार में उपलब्ध है। इस सॉफ्... सिमुलेटर प्रोग्राम में लिए IC-8051 अथवा इसको प्रयुक्त कर बने बोर्ड जिसे 8051-Board कहते हैं, की आवश्... नहीं होती है। PC (Personal Computer) में 8051 सॉफ्टवेयर सिमुलेटर प्रोग्राम को भर (Load) कर उपयोग... अपने प्रोग्राम को उस PC पर run कर सकता है। यह सॉफ्टवेयर, प्रोग्राम को अनुमति प्रदान करता है कि वह ... प्रोग्राम को लिख सके एवं उसमें त्रुटियों का पता लगा सके अर्थात् अपने प्रोग्राम को debug कर सके तथा ... निष्पादित (execute) कर सके। इसके अतिरिक्त यह सॉफ्टवेयर सिमुलेटर, प्रोग्रामकर्त्ता को यह अनुमति भी प्रदान ... है कि वह उस प्रोग्राम के परिणाम (Result) तथा अन्य विभिन्न रजिस्टरों के कन्टेन्ट्स (Contents) को देख सके।

प्रोग्राम का Simulator पर परीक्षण (Test) करते समय, यदि कोई त्रुटि(याँ) प्राप्त होती हैं तो उसे ... संशोधित करने के पश्चात् प्रोग्राम का पुन: सिमुलेटर पर परीक्षण करना चाहिए। जब प्रोग्राम में कोई त्रुटि प्राप्त ... अर्थात् जब सिमुलेशन (Simulation) द्वारा वांछित अनुक्रिया (Response)/परिणाम (Result) प्राप्त हो तो उस ... को माइक्रो-कन्ट्रोलर बोर्ड (Micr-Controller Board) में डाउनलोड (Download) कर लेते हैं। SIM-31 एवं ... क्रमश: सामान्य माइक्रो-कन्ट्रोलर तथा Linux Version of 8051 सिमुलेटर (Simulators) है।

प्रोग्राम को वास्तविक हार्ड-वेयर अर्थात् MCS-51 बोर्ड में कापी अथवा डाउन-लोड (download) करने ... पहले, यह अति आवश्यक है कि प्रोग्राम को सॉफ्टवेयर सिमुलेटर (Software Simulator) द्वारा परीक्षण (test) ... लिया जाए तथा वांछित परिणाम (result) न प्राप्त होने पर, उस प्रोग्राम में आवश्यक किंचित परिवर्तन (modify) ... के पश्चात् पुन: Software Simulator द्वारा परीक्षण करते हैं। इस उपरोक्त प्रक्रिया की पुनरावृत्ति करते रहते ... तक कि वांछित परिणाम न प्राप्त हो जाए। इस प्रकार, बहुत अधिक समय की बचत के साथ-साथ हार्डवेयर परीक्षण ... अधिक धन लगने से बचत होती है। एक बार प्रोग्राम का सिमुलेटर पर सत्यापन हो जाने पर अर्थात् परीक्षण में ... परिणाम प्राप्त होने के पश्चात् उस प्रोग्राम को ROM, EPROM अथवा EEROM में कापी कर लेते हैं।

अत: प्रोग्राम वांछित प्रक्रिया प्रदान करे तथा बार-बार प्रोग्राम को modify न करना पड़े, इसके लिए आवश्यक ... कि प्रारम्भ में ही सही प्रोग्राम लिखना चाहिए। इसके लिए MCS-51 के एसेम्बली प्रोग्रामिंग सांचे का ज्ञान ... आवश्यक है जिसका वर्णन आगे किया गया है।

## कम्पाइलर (Compiler)

एक कम्पाइलर एक Software Program होता है जो कि transform करता है High Level source code ... जो कि एक developer के द्वारा लिखा गया है एक high level programming language में उसे एक low level ... object code (binary code) machine language में, और जिसे कि processor के द्वारा आसानी से सांझा किया ... सके। इस प्रक्रिया को, जिसमें high-level programming को machine language में बदला जाता है, Compiler ... कहते हैं।

### Compiler के मुख्य भाग

Compiler के मुख्य रूप से दो मुख्य भाग होते हैं—

**(1) Analysis Phase :** इसमें एक Intermediate representation को create किया जाता है एक given ... source program से। इस phase के मुख्य भाग है—

(i) Lexical Analyzer

(ii) Syntax Analyzer

(iii) Semantic Analyzer

**(2) Synthesis Phase :** इसमें equivalent target program को create किया जाता है। Intermediate representation से। इस phase के मुख्य भाग है—

(i) Intermediate Code Generator

(ii) Code Generator

(iii) Code Optimizer



### कम्पाइलर के उपयोग

**(1) Scanning**—ये Scanner read करती है एक character एक समय में source code से और सभी characters का track रखता है जिससे ये पता चलता है कि कौन-सा character किस लाइन में उपलब्ध है।

**(2) Lexical Analysis :** Compiler Convert करता है Sequence of characters को जो कि source code में appear होते हैं उन्हें एक series of strings of characters में convert करता है।

**(3) Syntactic Analysis :** इस step में, Syntax analysis किया जाता है जिसमें Preprocessing involve होता है जो कि ये determine करता है कि क्या tokens जो कि create होता है Lexical analysis के दौरान वह proper order में है या नहीं।

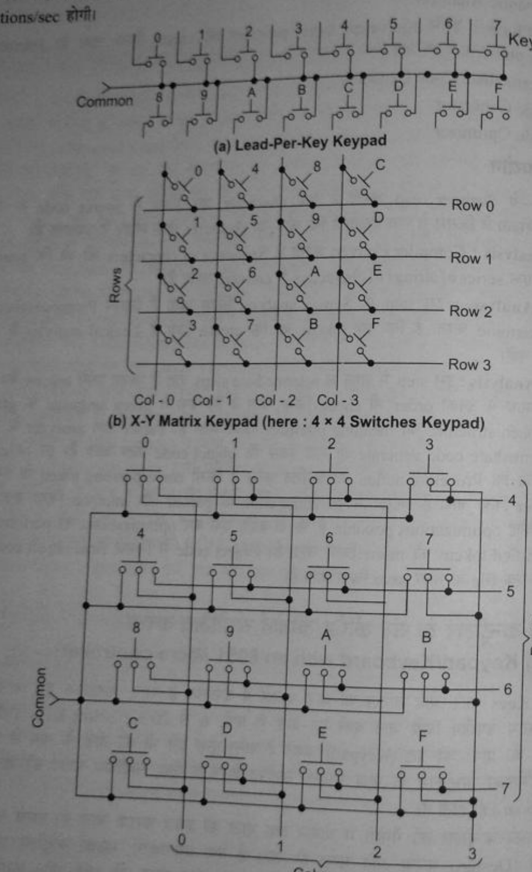
**(4) Semantic Analysis :** इस step में बहुत से intermediate steps होते है पहला इसमें tokens का structure check किया जाता है साथ में उनका order भी check किया जाता है कि क्या वे given language के grammar के अनुसार हैं या नहीं। Token structure का meaning interpret किया जाता है। Parser तथा analyzer के द्वारा जिससे कि finally एक intermediate code generate हो सके जिसे कि object code कहा जाता है। इन object code में instructions होते हैं जो कि Processor action को प्रदर्शित करते हैं किसी corresponding token के लिए जब उसे program में encounter किया जाता है आखिर में पूरा entire code को parsed और interpret किया जाता है यह चैक करने के लिए कि क्या कोई optimizations possible है भी या कही। एक बार optimizations को perform किया जाये, तब appropriate modified tokens को insert किया जाता है। Object code में जिससे final object code generate किया जा सके, जिसे कि एक file के भीतर save किया जाता है।

## 8051 माइक्रो-कन्ट्रोलर के संग कीपेड/कीबोर्ड संयोजित करना (Interfacing Keypad/Keyboard with an 8051 Micro-controller)

आजकल, 104 Keys वाले अति जटिल की-बोर्ड बाजार में उपलब्ध हैं परन्तु औद्योगिक एवं व्यापारिक-क्षेत्र में, माइक्रो-कन्ट्रोलर के साथ उपयोग किये जाने वाले की-बोर्ड में प्राय: 6 से 20 Keys/Soft Keys होती हैं। इन कम Keys वाले की-बोर्ड को प्राय: की-पेड (Keypad) कहते हैं परन्तु कुछ इसे भी की-बोर्ड के नाम से भी पुकारते हैं। मनुष्यों एवं कम्प्यूटरों/माइक्रो-कन्ट्रोलरों के मध्य सम्बन्ध स्थापित करने के लिए सर्वाधिक प्रयोग की जाने वाली इनपुट युक्ति Keypad/Keyboard ही होती है।

कीपेड द्वारा माइक्रो-कन्ट्रोलर की मैमोरी में प्रोग्राम तथा डाटा को प्रवेश कराया जाता है। इससे माइक्रो-कन्ट्रोलर आधारित अभिकल्पना (Design) करना अति सरल हो जाता है तथा प्रयोगकर्त्ता, माइक्रो-कन्ट्रोलर प्रणाली के साथ सरलता से सम्बन्ध स्थापित कर सकता है। उदाहरण के लिए, माना एक मोटर को ऑन-ऑफ करने के लिए तथा उसकी गति (speed) नियन्त्रित करने के लिए माइक्रो-कन्ट्रोलर का प्रयोग किया गया है। इस प्रक्रिया को इस प्रकार अभिकल्पित किया जा सकता है कि Keypad द्वारा प्रोसेसर में एक विशेष निर्देश प्रवेश कराने से मोटर गति करना प्रारम्भ कर दे एवं एक अन्य विशेष निर्देश फीड करने पर मोटर बन्द (Off) हो जाए। इसके अतिरिक्त Keypad द्वारा

विभिन्न डाटा को फीड करने पर मोटर की गति में नियन्त्रण सरलता से प्राप्त किया जा सकता है अर्थात् एक डाटा प्रवेश कराने पर यदि मोटर की गति *A* revolutions/sec होती है तो एक अन्य डाटा प्रवेश पर मोटर की ... revolutions/sec होगी।




(a) Lead-Per-Key Keypad

(b) X-Y Matrix Keypad (here : 4 × 4 Switches Keypad)

(c) Coded Keypad

चित्र 5.2 : Some Different Type Keypads/Keyboards

माइक्रो-कन्ट्रोलर/कम्प्यूटर में प्रोग्राम अथवा डाटा को कीबोर्ड/कीपेड द्वारा प्रवेश कराते समय निम्न सम्भावनाओं के प्रति सतर्कता रखना आवश्यक होता है—



- एक समय पर, एक से अधिक Keys प्रेस (अर्थात् ON) न हो।
- एक निश्चित समय से अधिक समय तक Key प्रेस अवस्था में न रहे।
- तीव्र गति से key, ऑन एवं ऑफ हो।
- Key की अवस्था परिवर्तित करते समय, उत्पन्न होने वाले सम्पर्क बाउन्स (Contact Bounce) का प्रवेश माइक्रो-कन्ट्रोलर में न होने पाये।

इन सभी स्थितियों का समाधान, हार्डवेयर अथवा सॉफ्टवेयर (Hardware or Software) द्वारा किया जा सकता है। यहाँ पर इन स्थितियों के समाधान के लिए, सॉफ्टवेयर के प्रयोग का वर्णन करेंगे क्योंकि यह विधि अधिक मूल्य-प्रभावशाली अर्थात् सस्ती है तथा इसमें कम रख-रखाव की आवश्यकता होती है तथा माइक्रो-कन्ट्रोलर का आकार छोटा (Compact) प्राप्त होता है।

## कीपेड समाकृति (Keypad Configurations)

व्यापारिक रूप से निर्मित किये जाने वाले 16 keys वाले कीपेड, निम्न तीन प्रकार के होते हैं—

- **प्रत्येक-की-लीड कीपेड** (Lead-Per-Key Keypad)—इस प्रकार के कीपेड में, प्रत्येक Key से एक संयोजक तार (Lead) बाहर निकली रहती है, जैसा कि चित्र 5.2 (a) में एक 16-Key वाले HEX Keypad को प्रदर्शित किया गया है।
- **X-Y मैट्रिक्स कीपेड** (X-Y Matrix Keypad)—कीपेड, प्रायः विभिन्न मैट्रिक्स संरूपों में उपलब्ध हैं, जैसा कि—(4 × 4), (2 × 8), (3 × 8) इत्यादि। चित्र 5.2 (b) में, सर्वाधिक उपयोग होने वाले 4 × 4 मैट्रिक्स युक्त, 16 Key वाले एक वर्ग कीपेड (Square Keypad) को प्रदर्शित किया गया है।
- **कोडेड कीपेड** (Coded Keypad)—इस प्रकार के Keypad की आन्तरिक वायरिंग को चित्र 5.2 (c) में प्रदर्शित किया गया है। इसके गुणों एवं स्थायित्व (durability) के कारण, इस प्रकार के Keypad का अधिक उपयोग किया जाता है परन्तु इस प्रकार के Keypad की आकृति में प्रायः अधिकतम 16-Keys होती हैं तथा यह सभी प्रकार के Keypads से अधिक महँगा (expensive) है।

इनमें से X—Y मैट्रिक्स में संयोजित कीपेड सर्वाधिक प्रसिद्ध है जबकि कीपेड की आकृति में 10 से अधिक Keys है। यदि मैट्रिक्स को एक वर्ग (Square) में व्यवस्थित किया जाए तो यह अधिक प्रभावशाली होता है। इसमें N-लीड दो (Rows अर्थात् X-अक्ष) तथा N-लीड कॉलम (Columns अर्थात् Y-अक्ष) में होती है जबकि Keys की संख्या $N \times N = N^2$ होती हैं। चित्र 5.2 (b) में प्रदर्शित 16-Keys वाले एक की पेड में 4-Rows तथा 4-Columns हैं।

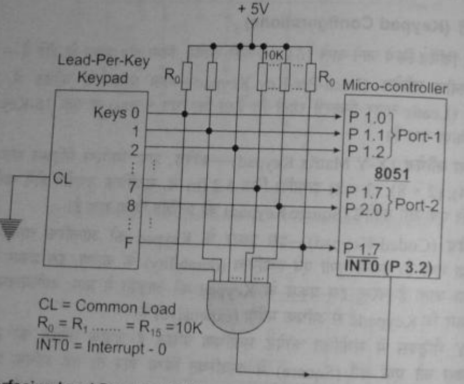
मूल रूप से कोडेड-कीपेड को टेलीफोनिक (Telephonic) अनुप्रयोग के लिए विकसित किया गया था जिससे, सम्पर्कों के परस्पर संयोजन के समय ध्वनि सिग्नल उत्पन्न किया जा सके। इस प्रकार के कीपेड में यदि एक-साथ, एक अधिक Keys प्रेस हो जाये तो उनका पता सरलता से हो जाता है।

## कीपेड के लिए प्रोग्राम (Programmes for Keypad)

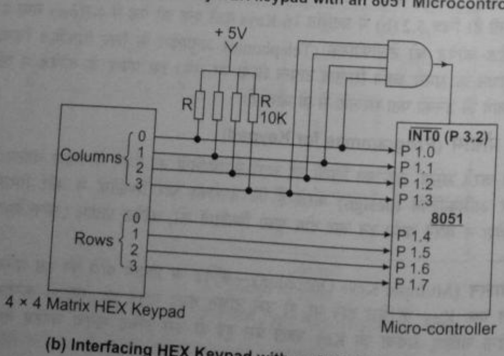
कीपेड को प्रयोग करते समय, उपरोक्त लिखी गई चारों सावधानियों को ध्यान में रखना चाहिए। इसके लिए कीपेड के प्रोग्राम को इस पर अभिकल्पित (design) करते हैं कि उपरोक्त चार स्थितियों में कोई विद्यमान होने पर डाटा, माइक्रो-कन्ट्रोलर में प्रवेश न करने पाये। इन चार दोष युक्त स्थितियों को, कीपेड प्रोग्राम (सॉफ्टवेयर) द्वारा निम्न प्रकार नियन्त्रित कर सकते हैं—

- **बहु की-प्रचालन** (Multiple Keys Operation)—कीपेड के प्रोग्राम ढाँचे को इस प्रकार अभिकल्पित करते हैं कि केवल एक Key के प्रेस होने पर ही उसे उचित डाटा मानते हुए, माइक्रो-कन्ट्रोलर में उस डाटा को प्रवेश होने देना चाहिए अथवा जो Key पहले प्रेस हुई हो उसे उचित मानना चाहिए तथा अन्य Key(s) को अस्वीकार करना चाहिए।

- **दबी स्थिति में की का रुकना** (Key Held in Pressed Situation)—वैध कीपेडड प्रोग्राम ढाँचे द्वारा Key के परिवर्तन को केवल मान्य डिबाउन्स विलम्ब समय (Debounce Delay Time) के पश्चात् ही स्वीकार करना चाहिए। एक बार किसी प्रेस्ड Key को स्वीकार करने के पश्चात् अन्य किसी Key को स्वीकार नहीं करना चाहिए, जब तक कि कुछ निश्चित समय के लिए सभी Keys, बिना प्रेस स्थिति अर्थात् सामान्य [illegible] स्थिति में न रहें।
- **सम्पर्क बाउन्स** (Contacts Bounce)—Key की दोनों परिस्थितियों (ON से OFF अथवा OFF से ON) [illegible] प्रोग्राम में विलम्ब-समय (Delay Time) का प्रयोग किया जाता है ताकि इस समय-अन्तराल में कोई अमान्य (invalid) डाटा, अथवा/और निर्देश, माइक्रो-कन्ट्रोलर में प्रवेश न कर सके।
- **तीव्र की प्रचालन** (Rapid Key Hit)—कीपेड के प्रोग्राम को इस प्रकार अभिकल्पित करना चाहिए कि—मनुष्य प्रतिक्रिया समय की अपेक्षा, Key अवलोकन करने (Scan) की दर अधिक होनी चाहिए।

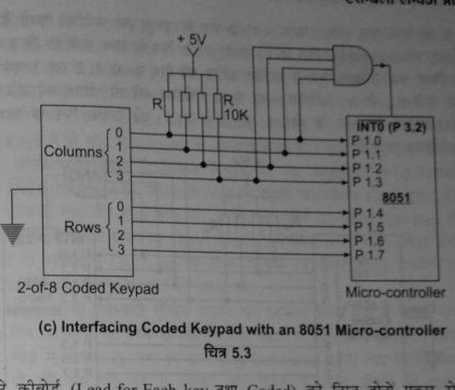



**(a) Interfacing Lead-Per-key HEX keypad with an 8051 Microcontroller**




**(b) Interfacing HEX Keypad with an 8051 Micro-controller**






**(c) Interfacing Coded Keypad with an 8051 Micro-controller**

चित्र 5.3

सामान्यतया छोटे कीबोर्ड (Lead-for-Each key तथा Coded) को निम्न दोनों प्रकार से प्रोग्रामिंग किया जा सकता है—

(i) इनकी कॉमन लीड को ग्राउन्ड करके, Key ढाँचे को समय-समय पर (Periodically) पढ़ते रहना।

(ii) सभी Keys के आउटपुट को AND गेट के इनपुटों पर संयोजित करके तथा AND गेट के आउटपुट को माइक्रो-कन्ट्रोलर की किसी एक इन्ट्रप्ट (Interupt) पिन $\overline{INT0}$ अथवा INT1 पर संयोजित किया जाता है। इससे, जब कोई Key प्रेस होती है तभी इन्ट्रप्ट सक्रिय होता है तथा कीपेड प्रोग्राम की स्केनिंग (Scanning) प्रारम्भ होती है। इस प्रकार, keypad को माइक्रो-कन्ट्रोलर के साथ संयोजित करने की विधि को चित्र 5.3 (a) में प्रदर्शित किया गया है।

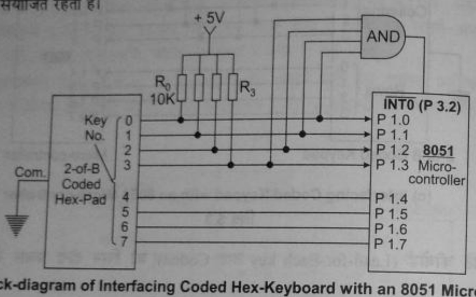
X-Y मैट्रिक्स कीबोर्ड में, प्रत्येक X-Row (0, 1, 2 अथवा 3) को क्रमशः शून्य अवस्था (निम्न विभव अवस्था) पर रखकर, Y-कॉलम में शून्य लीड (अर्थात् जो निम्न अवस्था में लीड होती है) का पता लगाते है जिससे कि मैट्रिक्स में प्रेस Key को ज्ञात किया जा सके। कीपेड प्रोग्राम नियन्त्रण में माइक्रो-कन्ट्रोलर के पोर्टों का उपयोग करके अथवा जटिल कीबोर्ड स्केनिंग परिपथ का प्रयोग करके X-Y स्केनिंग प्राप्त की जा सकती है। इस प्रकार की प्रोग्रामिंग के लिए 4 × 4 मैट्रिक्स कीपेड तथा Coded Keypad का संयोजन, 8051-माइक्रो-कन्ट्रोलर के साथ क्रमशः चित्र-5.3 (b एवं c) में प्रदर्शित किये गये हैं।

सभी प्रकार के कीबोर्डों के स्केनिंग प्रोग्राम को विकसित किया जा सकता है। उदाहरण के लिए, यहाँ पर हम एक कोडेड कीपेड के सरल प्रोग्राम का वर्णन करेंगे।

## 4 × 4 की युक्त कोडेड कीपेड के लिए इन्ट्रप्ट-संचालित प्रोग्राम (Interrupt Driven Programme for 4 × 4 Keys Coded Keypad)

यहाँ अब हम उदाहरण के रूप में, पूर्ण इन्ट्रप्ट ड्राइव छोटे (4 × 4 Keys युक्त) कोडेड कीपेड के प्रोग्राम का अध्ययन करेंगे। यह प्रोग्राम तब तक सक्रिय नहीं होता जब तक कि किसी एक Key को प्रेस न किया जाए। इसके लिए, 8051 माइक्रो-कन्ट्रोलर तथा 4 × 4 Key युक्त कोडेड कीपेड के मध्य संयोजन किये जाने वाले Hardware को चित्र-5.4 (a) में प्रदर्शित किया है। जैसा कि चित्र-5.4 (c) में भी प्रदर्शित किया गया है कि इस प्रकार के कीपेड में 9 टर्मिनल बाहर निकले रहते हैं—एक कॉमन पिन (Common Pin; Com) तथा दो निपल (Nipple—प्रत्येक निपल में 4 टर्मिनल होते हैं) ग्रुप-जिसमें से एक कॉलम ग्रुप (Column Group of Pin No. 0, 1, 2 and 3) तथा दूसरा रो-ग्रुप (Row Group-of Pin No. 4, 5, 6, and 7) है। कीपेड की कॉमन पिन (Com) को ग्राउण्ड करते हैं तथा कॉलम ग्रुप

की 0, 1, 2 एवं 3 को एक चार इनपुट वाले NAND गेट के इनपुट पर संयोजित करते है जिसके आउटपुट को 8051-माइक्रो-कन्ट्रोलर की Interrupt-0 पिन $\overline{INT0}$ अर्थात् P3.2 पिन के साथ संयोजित किया गया है, जैसा कि चित्र 5.3 (a) में प्रदर्शित किया गया है। इस चित्र में प्रदर्शित कीपेड की पिन नं० 0, 1, 2 एवं 3 एवं पावर सप्लाई के साथ संयोजित 10 k मान वाले $R_0$ से $R_3$ प्रतिरोध अगर न भी संयोजित करें तो परिपथ की प्रक्रिया में कोई अन्तर नहीं आयेगा क्योंकि 8051-माइक्रो-कन्ट्रोलर के पोर्ट-1 (P1.0 से P1.7) की प्रत्येक पिन के साथ अन्दर ही पुल-अप (Pull-up) प्रतिरोध संयोजित रहता है।




चित्र 5.4 (a) : Block-diagram of Interfacing Coded Hex-Keyboard with an 8051 Micro-controller for scanning the Code Key Program

| Key Pressed | Status of Key Word | | | Coded Key No. in Program = HN - 77 |
|---|---|---|---|---|
| | Rows | Columns | Number in Hex-No. System (HN) | |
| | 7 6 5 4 | 3 2 1 0 | | |
| 0 | 1 1 1 0 | 1 1 1 0 | E E | 77 |
| 1 | 1 1 1 0 | 1 1 0 1 | E D | 76 |
| 2 | 1 1 1 0 | 1 0 1 1 | E B | 74 |
| 3 | 1 1 1 0 | 0 1 1 1 | E 7 | 70 |
| 4 | 1 1 0 1 | 1 1 1 0 | D E | 67 |
| 5 | 1 1 0 1 | 1 1 0 1 | D D | 66 |
| 6 | 1 1 0 1 | 1 0 1 1 | D B | 64 |
| 7 | 1 1 0 1 | 0 1 1 1 | D 7 | 60 |
| 8 | 1 0 1 1 | 1 1 1 0 | B E | 47 |
| 9 | 1 0 1 1 | 1 1 0 1 | B D | 46 |
| A | 1 0 1 1 | 1 0 1 1 | B B | 44 |
| B | 1 0 1 1 | 0 1 1 1 | B 7 | 40 |
| C | 0 1 1 1 | 1 1 1 0 | 7 E | 07 |
| D | 0 1 1 1 | 1 1 0 1 | 7 D | 06 |
| E | 0 1 1 1 | 1 0 1 1 | 7 B | 04 |
| F | 0 1 1 1 | 0 1 1 1 | 7 7 | 00 |

चित्र 5.4 (b) : A Byte Address for the Key Pressed in Fig. (a) and Key codes in the key Scan Programme



इस Hex कोडेड कीपेड की किसी एक Key को प्रेस करने से, कीपेड के कॉलम ग्रुप (पिन नं० 0, 1, 2 या 3) की केवल एक पिन ग्राउण्ड (0-अवस्था) में रहती है तथा 8 पिनों (0 से 7 पिनें) में से केवल 2-पिनें 0-अवस्था में रहती हैं। यदि किसी कारणवश, एक से अधिक Keys प्रेस हो जायें तो कीपेड की 3 या अधिक पिनें 0-अवस्था में होती है जोकि एक अवैध (invalid) प्रक्रिया है। कीपेड के प्रोग्राम को इस प्रकार अभिकल्पित करते है कि इस स्थिति में (एक से अधिक Keys प्रेस होने पर) किसी भी Key के अंक, माइक्रो-कन्ट्रोलर में प्रवेश नहीं करने पाते है।

जब कीपेड की 16 Keys में से कोई Key(s) प्रेस हो तो कीपेड की 0-3 पिनों में से एक पिन अवश्य 0-अवस्था में होने से AND गेट असक्रिय हो जाता है जिससे AND गेट के आउटपुट पर 0-अवस्था प्राप्त होने से माइक्रो-कन्ट्रोलर की पिन $\overline{INT0}$ (Interrupt-0) सक्रिय हो जाता है। इसके फलस्वरूप CPU द्वारा, कीपेड प्रोग्राम स्कैन (Scan) होना प्रारम्भ हो जाता है तथा वैध Key होने पर अर्थात् एक समय पर केवल एक Key प्रेस होने पर, उस Key का अंक माइक्रो-कन्ट्रोलर में उचित समय पर (Keys Debounce समय के पश्चात्) प्रवेश करके स्टोर हो जाता है। 8051-माइक्रो-कन्ट्रोलर में अधिकतम् 69.53 ms का समय-विलम्ब (Time-Delay) उत्पन्न किया जा सकता है जबकि क्लॉक-आवृत्ति (Clock Frequency) का मान 12 MHz हो।

अग्र Hex Coded Keypad के एसेम्बली लैंग्वेज प्रोग्राम को प्रदर्शित किया गया है जोकि माइक्रो-कन्ट्रोलर की इन्ट्रप्ट-0 (Interrupt-0) पिन पर पल्स के उच्च-से-निम्न संक्रांति-काल (Transition-period) प्राप्त होने पर सक्रिय हो जाता है। इसमें इन्ट्रप्ट मोड में, टाइमर $T_0$ तथा $T_1$ द्वारा Key Debounce समय तथा डिले समय (Delay Time) उत्पन्न करता है। कौन-सी की प्रेस की गई है इसको ज्ञात करने के लिए प्रोग्राम में एक की-प्रेस-तालिका (Key-Press-Table) का प्रयोग किया गया है।

## सप्त खण्ड प्रदर्शक का माइक्रो-कन्ट्रोलर के साथ संयोजन (Interfacing 7-segment Display with Microcontroller)

माइक्रो-कन्ट्रोलर के आउटपुट को प्रदर्शित करने के लिए माइक्रो-कन्ट्रोलर के आउटपुट पर 7-सैग्मेन्ट को संयोजित करना अधिक व्यापक विधि है। LED डिस्प्ले के प्रयोग को उन स्थानों पर प्राथमिकता दी जाती है जहाँ-पावर व्यय अधिक समस्या न हो अन्यथा LCD डिस्प्ले अथवा ऑरगेनिक प्रकाश उत्सर्जित डायोड (Organic Light Emitting Diodes; OLEDs) प्रयुक्त डिस्प्ले का प्रयोग किया जाता है।

OLED, एक इलैक्ट्रॉनिक युक्ति है जो दो चालकों (Conductors) के मध्य पतली ऑरगेनिक फिल्म (thin organic films) को श्रेणी में रखकर निर्मित की जाती है। जब इसमें विद्युत धारा प्रवाहित की जाती है तो चमकदार प्रकाश उत्पन्न होता है। OLEDs भार में हल्के (light in weight), टिकाऊ (Durable), पावर दक्ष (Power Efficient अर्थात् कम पावर व्यय करने वाले), सस्ते तथा एक स्थान से दूसरे स्थान तक ले जाने मे आदर्श युक्ति है। वैज्ञानिकों का विश्वास है कि वर्तमान टैक्रिक द्वारा निर्मित किये जा रहे डिस्प्ले युक्तियों के स्थान पर अपने गुणों के फलस्वरूप OLEDs युक्ति डिस्प्ले व्यापक रूप में प्रयोग किये जायेंगे।

7-सैग्मेन्ट डिस्प्ले को सीधे 8051-माइक्रो-कन्ट्रोलर के साथ संयोजित करने की एक विधि को चित्र 5.5 में प्रदर्शित किया गया है। इसमें Hexadecimal अथवा Decimal संख्या के जिस अंक को 7-सैग्मेन्ट डिस्प्ले पर प्रकाशित करना होता है, इसके बाइनरी कोड को माइक्रो-कन्ट्रोलर के प्रोग्राम द्वारा स्वत: उत्पन्न किया जाता है। इस आरेख में 8051-माइक्रो-कन्ट्रोलर के पोर्ट-1 (P1) एवं पोर्ट-3 की चार पिनों (P3-0 से P3.3 तक) के साथ चार 7-सैग्मेन्ट डिस्प्ले संयोजित (connect) किये गये हैं। अत: इस प्रकार चार अंकों की संख्या को प्रदर्शित किया जा सकता है। इसमें सभी चारों 7-Segment Display, कॉमन कैथोड प्रकार के डिस्प्ले प्रयुक्त किये गये है, अत: उसी 7-सैग्मेन्ट डिस्प्ले के Segments प्रकाशित होंगे, जिस डिस्प्ले का कॉमन कैथोड टर्मिनल ग्राउण्ड (अर्थात् 0-अवस्था में) होगा।

इस प्रकार, एक से अधिक डिस्प्ले युक्तियों को एक माइक्रो-कन्ट्रोलर के आउटपुट पर संयोजित करने को डिस्प्ले की समय-बहुविध (Time-multiplexing of Displays) कहते है। इस विधि द्वारा 7-Segment Displays को संयोजन

करने से माइक्रो-कन्ट्रोलर की कम पिनें प्रयोग होती है। इस उदाहरण (चित्र 5.5) में, आवश्यक पिनों की संख्या (पोर्ट-1) + 4 (पोर्ट 3 की 4 पिनें, P3.0 – P3.3) = 12 है। इसकी प्रक्रिया को उदाहरण द्वारा सरलता से स्पष्ट किया जा सकता है। माना कि हमें चित्र 5.5 में संयोजित 7-Segment Displays पर संख्या 9.179 प्रदर्शित करनी है, अतः सर्वप्रथम, 5 अंक की संख्या के बाइनरी कोड को पोर्ट-1 पर लोड करते हैं। चूँकि इस 7-Segment Display में Decimal Point भी Glow करना चाहिए अतः h-Segment के एनोड पर भी 1-अवस्था प्राप्त होने के लिए $Q_h$-ट्रांजिस्टर के बेस पर भी 0-अवस्था प्राप्त होनी चाहिए, अतः तालिका 9.6.2 के अनुसार, 5 अंक के लिए कोड 0010010 (अर्थात् 92H) के स्थान पर बाइनरी कोड 00010010 (अर्थात् 12 H) का प्रयोग करते हैं। PNP ट्रांजिस्टर $Q_h$, $Q_g$, $Q_f$, $Q_d$, $Q_c$ एवं $Q_a$ के बेसों (Bases) पर 0-अवस्था मिलने से वे ऑन हो जाते हैं जिसके फलस्वरूप क्रमशः $h, g, f, d, c$ एवं $a$ सैग्मेन्ट के एनोडों पर 1-अवस्था (अर्थात् उच्च विभव अवस्था) प्राप्त होती है। अब यदि 7-सैग्मेन्ट डिस्प्ले $7SD_1$ के कॉमन कैथोड (Common Cathode; CC) के साथ संयोजित ट्रांजिस्टर $Q_1$ के बेस पर 1-अवस्था प्रदान की जाए तो $7SD_1$ डिस्प्ले की $h, g, f, d, c$ एवं $a$ सैग्मेन्ट, प्रकाश उत्सर्जित करने लगते हैं जिससे 5-संख्या तथा उसके पश्चात् दशमलव बिन्दु (अर्थात् 9.) दृष्टिगोचर होता है।

इसके पश्चात् तालिका 5.5 से अनुसार, 1-संख्या के बाइनरी-कोड-11111001 (अर्थात् F9H) को पोर्ट-1 पर लोड करते हैं तथा पोर्ट-3 की पिन P3.1 को 1-अवस्था में करते हैं इससे $7SD_2$ डिस्प्ले की $b$ एवं $c$ सैग्मेन्ट प्रकाशित हो जाती हैं जिससे 1-संख्या दृष्टिगोचर होती है। इसी प्रकार, $7SD_3$ एवं $7SD_4$ की डिस्प्ले में क्रमशः 7-संख्या एवं 9-संख्या के बाइनरी कोड प्रयुक्त कर, माइक्रो-कन्ट्रोलर पिन P3.2 एवं P3.3 द्वारा क्रमशः ट्रांजिस्टर $Q_3$ एवं $Q_4$ को ऑन करते हैं इस प्रकार, $7SD_3$ एवं $7SD_4$ डिस्प्ले पर क्रमशः 7 एवं 9 संख्या दृष्टिगोचर होती है। तालिका 5.1 में 9.179 संख्या प्रकाशित करने के लिए प्रयुक्त करने वाले बाइनरी कोड को पुनः प्रदर्शित किया गया है।

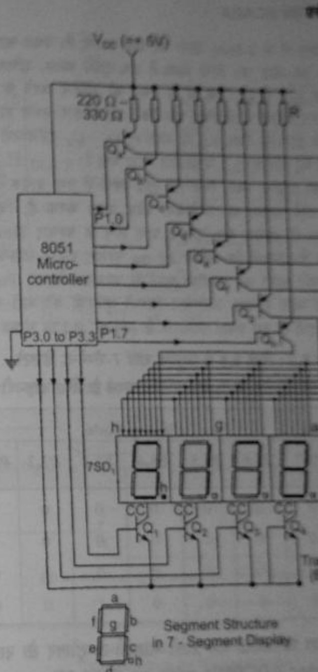
**तालिका 5.1 : चित्र 5.5 में 9.179 संख्या प्रकाशित करने के लिए प्रयुक्त कोड**

| No. | 7-Segment Display Name | Code for Display (Binary) | | | | | | | | HEX Code (H) |
|---|---|---|---|---|---|---|---|---|---|---|
| | | h | g | f | e | d | c | b | a | |
| 5 | $7SD_1$ | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 12 |
| 1 | $7SD_2$ | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | F9 |
| 7 | $7SD_3$ | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | F8 |
| 9 | $7SD_4$ | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 98 |

चूँकि चित्र 5.5 में संयोजित चार 7-Segment Displays ($7SD_1$ से $7SD_4$ तक) में से, एक समय पर केवल एक डिस्प्ले को ही डाटा (आवश्यक संख्या के बाइनरी कोड) प्राप्त होता है परन्तु 9.179 पूर्ण संख्या सदैव ही दृष्टिगोचर होनी चाहिए। इसके लिए हम मानव के **दृष्टि दोष** (Sight Effect) अर्थात् दृष्टि भ्रम का सहारा लेते हैं। इसके अनुसर, यदि किसी स्थान पर (यहाँ Segment में) पर कुछ छड़ों (μs रेन्ज में) के लिए प्रकाश उत्पन्न किया जाये तथा 10 ms के अन्दर पुनः उसी स्थान (यहाँ उन्हीं Segments में) प्रकाश उत्पन्न किया जाए तथा इसी प्रकार, उस स्थान पर प्रकाश उत्पन्न करने की बार-बार पुनरावृत्ति की जाए जबकि प्रत्येक बार प्रकाश उत्पन्न करने के मध्य समय अन्तर 10 ms या इससे कम रख जाए तो मानव को वह स्थान (यहाँ वे Segments) प्रकाशित दृष्टिगोचर होता रहता है जब तक कि उस स्थान पर, 10 ms अथवा इससे कम समय अन्तराल पर, कुछ छड़ों (μs रेन्ज में) में प्रकाश उत्पन्न करने की प्रक्रिया की पुनरावृत्ति होती रहती है।

अतः चित्र 5.5 में प्रयुक्त चारों 7-सैग्मेन्ट डिस्प्ले में से प्रत्येक डिस्प्ले को 10 ms के अन्दर ही पुनः उसी संख्या का बाइनरी कोड प्रयुक्त करते हैं जो संख्या पहले से उसमें प्रकाशित की गई है। अतः एक 7-Segment Display (माना $7SD_1$) एवं दूसरी अग्र 7-सैग्मेन्ट डिस्प्ले को सक्रिय करने के मध्य समय (≤ (10 ms ÷ number of 7-segment Displays (यहाँ 4 है) होना चाहिए। अतः चार 7-सैग्मेन्ट डिस्प्ले प्रयुक्त परिपथ में, एक डिस्प्ले एवं उसकी अग्र डिस्प्ले में डाटा प्रयुक्त करने का अधिकतम समय-अन्तर (10 ÷ ms 4 = 2.5 ms) 2.5 ms होना चाहिए।






**चित्र 5.5 : Interface Time-Multiplexed Common Cathode Seven-Segment Displays with 8051-Microcontroller**

चित्र 5.5 में $Q_a - Q_h$ तथा $Q_1 - Q_4$ ट्रांजिस्टरों का प्रयोग परिपथ में किया गया है जोकि 7-सैग्मेन्ट डिस्प्ले के प्रत्येक सैग्मेन्ट को लगभग 10 mA धारा प्रदान कर सके तथा प्रत्येक डिस्प्ले के कॉमन कैथोड पर प्रवाहित होने वाली लगभग 70 mA धारा भी ट्रांजिस्टर के थ्रू प्रवाहित हो सके तथा 8051-माइक्रो-कन्ट्रोलर का आउटपुट लोड न करने पाये।

चित्र 5.5 में प्रदर्शित परिपथ में प्रयुक्त 7-सैग्मेन्ट डिस्प्ले युक्तियों को श्रेणीक्रम में ड्राइव (Drive) करने के लिए, Programme # 9.6.1 का प्रयोग किया जा सकता है। यह प्रोग्राम डेसीमल बिन्दु सहित चार अंकों की संख्या 9.179 को प्रदर्शित करता है जिसके अंकों के 7-सैग्मेन्ट कोड, तालिका 5.1 के अनुसार, क्रमशः 12H, F9H, F8H एवं 98H है तथा जिन्हें माइक्रो कन्ट्रोलर के आन्तरिक RAM में क्रमशः 40, 42, 44 एवं 46H लोकेशन पर स्टोर किया गया है। इस मुख्य प्रोग्राम में साधारणतः टाइमर-0 को Mode-1 में प्रयोग किया गया है तथा 2.5 ms पश्चात्, टाइमर द्वारा सॉफ्ट इन्ट्रप्ट-0 प्राप्त करने के लिए अर्थात् प्रोग्राम काउण्टर (PC) को प्रोग्राम लोकेशन एड्रेस 000BH पर जम्प कराने के लिए, टाइमर-0 में संख्या-F2FBH लोड करते हैं। जब टाइमर-0 का इन्ट्रप्ट (Interrupt) एक बार सक्रिय होता है तो

वह 40, 42, 44 अथवा 46H लोकेशन में से 2-Byte डाटा को प्राप्त करता है। प्रथम बाइट (8-बिट) तो 7-Segment डिस्प्ले कोड होता है जोकि पोर्ट-1 पर लोड कर दिया जाता है तथ दूसरी बाइट, उचित 7-सैग्मेन्ट डिस्प्ले के कॉमन कैथोड पर संयोजित NPN ट्रांजिस्टर ($Q_1$ से $Q_4$ तक में से एक) को सक्रिय करने के लिए डाटा होता है, जिसकी 8-बिटों में से जिस बिट (P3.0 से P3.3 तक में से जिस एक बिट) को सक्रिय करना होता है, वह बिट 1-अवस्था में होती है तथा अन्य बिटें 0-अवस्था में होती हैं। चित्र 5.5 में संयोजित $Q_1 - Q_4$ ट्रांजिस्टरों में से, एक समय पर केवल एक ट्रांजिस्टर को ऑन करने के कोड को तालिका 5.2 में प्रदर्शित किया गया है।

मुख्य प्रोग्राम द्वारा, टाइमर-0 की प्रक्रिया स्वत: नियन्त्रित होती रहती है तथा प्रत्येक 2.5 ms के पश्चात् इन्ट्रप्ट-0 द्वारा अग्र डिस्प्ले को अवश्य पुन: उसी संख्या का 7-सैग्मेन्ट कोड प्रदान करता है, जिस कोड द्वारा पहले से वह 7-Segment Display प्रकाशित रहता है। प्रत्येक बार, इन्ट्रप्ट प्राप्त होने के पश्चात् Timer-0 के रजिस्टर 7H1 एवं 7H2 को लोड करना आवश्यक होता है। जिससे कि प्रत्येक 2.5 ms पश्चात् इन्ट्रप्ट सक्रिय हो सके। 2.5 ms अन्तराल पर बारी-बारी 7-सैग्मेन्ट डिस्प्ले के चारों अंकों को प्रकाशित करने के पश्चात्, पोइन्टर (Pointer) में पुन: Ist अंक का Address प्रोग्राम द्वारा स्वत: लोड हो जाता है तथा 7-सैग्मेन्ट डिस्प्ले युक्तियों को बारी-बारी से आधुनिक (update) बनाने की प्रक्रिया की पुनरावृत्ति होती रहती है। इस प्रकार उदाहरण के अनुसार, 9.179 संख्या दृष्टिगोचर होती रहती है।

**तालिका 5.2 : चित्र 5.5 में प्रयुक्त चार 7-सैग्मेन्ट डिस्प्ले में से एवके कॉमन कैथोड (CC) को सक्रिय करने के लिए बाइनरी कोड**

| Active Transistor | Active CC of 7-Segment Display | Binary Code | | | | | | | | HEX Code (H) |
|---|---|---|---|---|---|---|---|---|---|---|
| | | P3.7 | P3.6 | P3.5 | P3.4 | P3.3 | P3.2 | P3.1 | P3.0 | |
| $Q_1$ | $7SD_1$ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 01 |
| $Q_2$ | $7SD_2$ | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 02 |
| $Q_3$ | $7SD_3$ | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 04 |
| $Q_4$ | $7SD_4$ | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 08 |

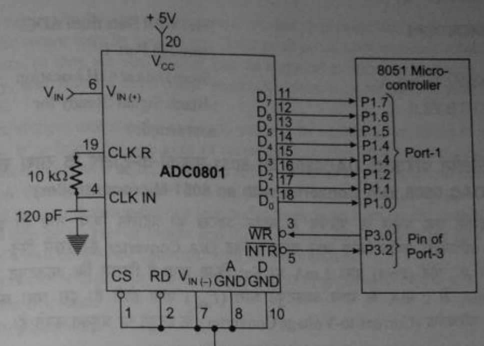
## ADC0801 एनेलॉग-से-डिजिटल परिवर्तक का 8051-माइक्रो-कन्ट्रोलर के साथ संयोजन (Interfacing A/D converter ADC0801 with an 8051-Microcontroller)

इसके एक प्रकार के संयोजन परिपथ को चित्र 5.6 में प्रदर्शित किया गया है तथा इसके एक नियन्त्रक प्रोग्राम को प्रोग्राम 5.2 में प्रदर्शित किया गया है। यह प्रोग्राम निम्न प्रक्रिया प्रदान करता है—

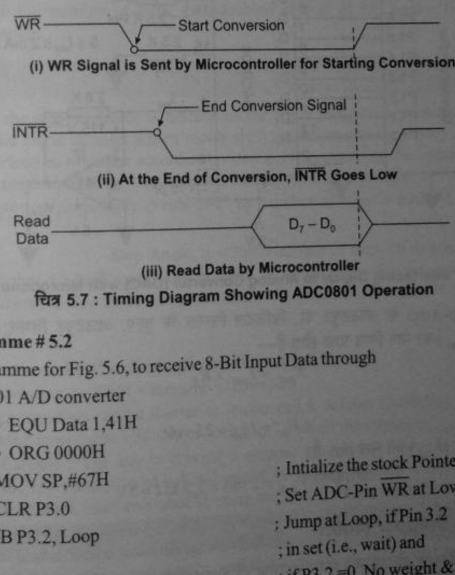
सर्वप्रथम माइक्रो-कन्ट्रोलर, पोर्ट-3 की पिन-P3.0 द्वारा A/D परिवर्तक की पिन $\overline{W/R}$ पर 0-अवस्था (निम्न विभव) सिग्नल प्रदान करता है। इस सिग्नल के प्राप्त होने पर A/D परिवर्तक, उसके इनपुट (पिन-6) पर विद्यमान विभव आयाम को, उसके तुल्य डिजिटल सिग्नल में परिवर्तन करना प्रारम्भ (start) कर देता है। जब ADC द्वारा एनेलॉग सिग्नल को डिजिटल सिग्नल में परिवर्तन करने की प्रक्रिया पूर्ण हो जाती है तो ADC की INTER (पिन-5), 0-अवस्था (High-to-Low Start) में पहुँच जाती है। इससे माइक्रो-कन्ट्रोलर की पिन P3.2 पर सिग्नल प्राप्त होता है कि Data तैयार है। माइक्रो-कन्ट्रोलर, इस डाटा को पोर्ट-1 द्वारा पढ़ने के पश्चात् इस डाटा को अपनी RAM की लोकेशन 41H में स्टोर कर देता है। अब माइक्रो-कन्ट्रोलर, ADC की पिन-*WR* (3No. पिन) पर 1-अवस्था प्रदान करता है जिससे कि ADC, न्यू डाटा पर कार्य करने के लिए तैयार हो जाये।

इस प्रोग्राम द्वारा परिचालित परिपथ (चित्र 5.6) के समय-आरेख को चित्र 5.7 में प्रदर्शित किया गया है।






चित्र 5.6 : Interfacing Analog to Digital Converter (ADC) with microcontroller



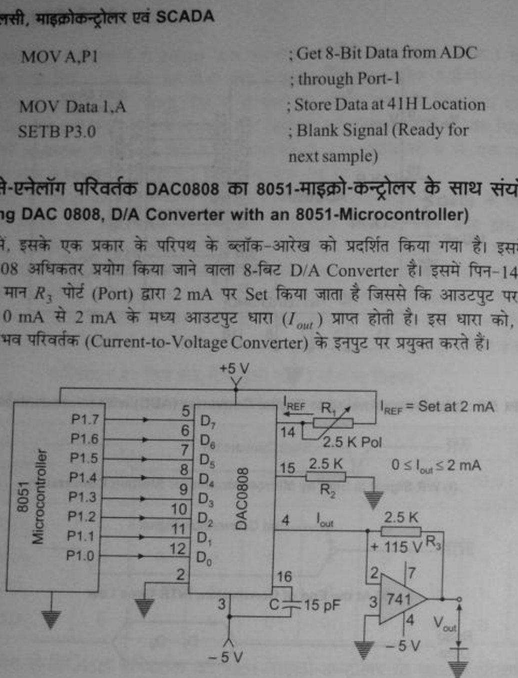

चित्र 5.7 : Timing Diagram Showing ADC0801 Operation

```
; Programme # 5.2
; A Programme for Fig. 5.6, to receive 8-Bit Input Data through
; ADC0801 A/D converter
        • EQU Data 1,41H
        • ORG 0000H
        MOV SP,#67H        ; Intialize the stock Pointer
        CLR P3.0           ; Set ADC-Pin WR at Low
Loop :  JB P3.2, Loop      ; Jump at Loop, if Pin 3.2
                           ; in set (i.e., wait) and
                           ; if P3.2 =0, No weight & go
                           ; at next Instruction
```

```
MOV A,P1          ; Get 8-Bit Data from ADC
                  ; through Port-1
MOV Data 1,A      ; Store Data at 41H Location
SETB P3.0         ; Blank Signal (Ready for
                    next sample)
```

## डिजिटल-से-एनेलॉग परिवर्तक DAC0808 का 8051-माइक्रो-कन्ट्रोलर के साथ संयोजन (Interfacing DAC 0808, D/A Converter with an 8051-Microcontroller)

चित्र 5.8 में, इसके एक प्रकार के परिपथ के ब्लॉक-आरेख को प्रदर्शित किया गया है। इसमें संयोजित D/A परिवर्तक DA 0808 अधिकतर प्रयोग किया जाने वाला 8-बिट D/A Converter है। इसमें पिन-14 पर Reference धारा ($I_{REF}$) का मान $R_3$ पोर्ट (Port) द्वारा 2 mA पर Set किया जाता है जिससे कि आउटपुट पर, इनपुट डिजिटल सिग्नल के तुल्य, 0 mA से 2 mA के मध्य आउटपुट धारा ($I_{out}$) प्राप्त होती है। इस धारा को, 741 Op. Amp प्रयुक्त धारा-से-विभव परिवर्तक (Current-to-Voltage Converter) के इनपुट पर प्रयुक्त करते हैं।




चित्र 5.8 : Interfacing Digital to Analog Converter (DAC) with Microcontroller

इस प्रकार 741-Op-Amp के आउटपुट पर, डिजिटल सिग्नल के तुल्य, आउटपुट विभव ($V_{out}$) प्राप्त होता है। इस आउटपुट वोल्टेज ($V_{out}$) का मान निम्न प्राप्त होता है—

$$V_{out} = I_{out} \times R_3$$

इस परिपथ में—

$$V_{ut} = I_{out} \times 25 \text{ volt}$$

जहाँ, आउटपुट धारा ($I_{out}$) का मान mA है।

अतः अधिकतम $V_{out} = 2 \text{ mA} \times 2.5 \text{ k}\Omega = 5\text{V}$

## Interfacing of Stepper Motor

Before going to know about the interfacing of stepper motor with 8051, let's know about few hardware details of stepper motor first.

Stepping motors can be viewed as electric motors without commutators. All of the commutation



must be handled externally by the motor controller, and typically, the motors and controllers are designed so that the motor may be held in any fixed position as well as being rotated one way or the other. Most steppers, as they are also known, can be steppted at audio frequencies, allowing them to spin quite quickly, and with an appropriate controller, they may be started and stopped in a very controlled manner.

Of all motors, stepper motor is the easiest to control. Its handling simplicity is really hard to deny - all there is to do is to bring the sequence of rectangle impulses to one input of step controller and direction information to another input. Direction information is very simple and comes down to "left" for logical one on that pin and "right" for logical zero. Motor control is also very simple-every impulse makes the motor operating for one step and if there is no impulse the motor won't start. Pause between impulses can be shorter or longer and it defines revolution rate. This rate cannot be infinite because the motor won't be able to "catch up" with all the impulses.

Stepper motors can be driven in two different patterns or sequences. Namely,

(i) Full Step Sequence

(ii) Half Step Sequence

In the **full step sequence,** two coils are energized at the same time and motor shaft rotates.

In **Half mode step sequence,** motor step angle reduces to half the angle in full mode. So the angular resolution is also increased *i.e.*, it becomes double the angular resolution in full mode. Also in half mode sequence the number of steps gets doubled as that of full mode. Half mode is usually preferred over full mode.

## Step Angle

Step angle of the stepper motor is defined as the angle trasversed by the motor in one step, in other words, it is the angle through which motor shaft rotates in one step. Step angle is different for different motors. Selection of motor according to step angle depends on the application, simply if you require small increments in rotation, choose motor having smaller step angle.

To calculate step angle, simply divide 360° by number of steps a motor takes to complete one revolution.

$$\text{Step Angle, } \phi = 360°/\text{Number of steps in degree}$$

As we know that in half mode, the number of steps taken by the motor to complete one revolution gets doubled, so step angle reduces to half.

### Steps/Second

The relation between RPM and steps per sec. is given by :

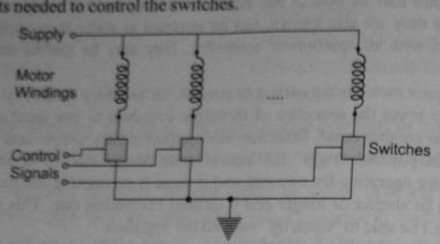
Steps or impulses/sec. = (RPM × Steps/revolution)/60

Pause between impulses can be shorter or longer and it defines revolution rate. This rate cannot be infinite because the motor won't be able to "catch up" with all the impulses.

**Interfacing Circuit :** The key to driving a stepper is realizing how the motor is constructed. Figure 5.9 shows the representation of a 4 coil motor, so named because 4 coils are used to cause the revolution of the dirve shaft. Each coil must be energized in the correct order for the motor to spin.

The circuitry shown in figure 5.9 is centered on a single issue, switching the current in each motor winding on and off, and controlling its direction. The circuitry is connected directly to the motor windings and the motor power supply, and this circuitry is controlled by a digital system that determines when the switches are turned on or off. A control unit is responsible for providing the control signals to open and close the switches at the appropriate times in order to spin the motors. The

control unit is commonly a computer or programmable interface controller, with software d... generating the outputs needed to control the switches.

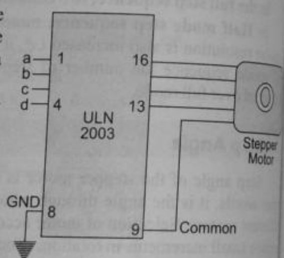



चित्र 5.9 : 4-Coil Stepper Motor

## Interfacing with 8051

To cause the stepper to rotate, we have to send a pulse to each coil in turn. The 8051 does not have sufficient drive capability on its output to drive each coil, so there are a number of ways to drive a stepper.

Stepper motors are usually controlled by transistor or dirver IC like ULN2003. Driving current for each coil is then needed about 60 mA at + 5V supply. A Darlington transistor array, ULN2003 is used to increase driving capacity of the 8051 chip. Four resistors help the 8051 to provide more sourcing current from the + 5V supply.




चित्र 5.10 : Interfacing circuit

The Pins 1 to 4 of ULN2003 is connected to the microcontroller pins. Through Pins 13 to 16 of ULN 2003, Stepper motor draws the control sequence for the sake of rotation. Pin 9 is common and 8 is ground.

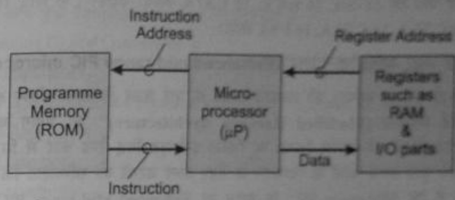
(i) **MOV** stepper, # 03H
(ii) **ACALL** delay
(iii) **MOV** stepper, # 01H
(iv) **ACALL** delay
(v) **MOV** stepper, # 00H
(vi) **ACALL** delay
(vii) **MOV** stepper, # 02H
(viii) **ACALL** delay
(ix) **SJMP** main

## पीआईसी माइक्रोकन्ट्रोलर्स का परिचय

PIC and PICmicro परिवाणी शब्द "Microchip Technology के Trademarks शब्द है। General Instruments के विभाग **"Microchip Technology"** ने प्रारमभ में अपने द्वारा निर्मित "Single-chip micro controllers" का नाम कराया। **PIC (Programmable Interface Controller)** दिया परन्तु कुछ समय पश्चात्



इसने परिवाणी शब्द (acronyms) PIV का नामकरण "Programmable Intelligent Computer" कर दिया। इस जिस संरचना का उपयोग किया गया उसे "Modified Harvard Architecture" नाम दिया गया। PIC की बेसिक संरचना के ब्लॉक आरेख को चित्र 5.11 में प्रदर्शित किया गया। इस हार्वर्ड संरचना में, माइक्रोप्रोसेसर (CPU) में निर्देशों (Instructions) को Programme Memory से लाने (Fetch) के लिए जिन बसों (Buses) का प्रयोग किया जाता है वे बसें Accessing Variables के लिए प्रयोग की जाने वाली बसों से अलग होती हैं जैसा कि चित्र 5.11 में प्रदर्शित किया गया है। सामान्यत: अन्य माइक्रोकन्ट्रोलरों में Programme Memory एवं Accessing Variables (*i.e.*, Data) के लिए अलग-अलग बसों का उपयोग नहीं किया जाता है। अत: CPU द्वारा Programme Memory से अब निर्देश ग्रहण करने के पहले प्रोग्राम डाटा की Read/Write एवं Input/Output प्रक्रियाओं के समाप्त होने का इन्तजार करना पड़ता है। इससे माइक्रोकन्ट्रोलर की चाल (speed) में कमी आती है।




चित्र 5.11 : Harvard Architecture of PIC Microcontroller

परन्तु PIC की Harvard Architecture होने के कारण बिना इन्तजार किये प्रत्येक चक्र (Cycle) में निर्देशों को Fetch किया जा सकता है। इसमें, जिस चक्र मे कोई निर्देश फैच होता है उसी चक्र में ही CPU, इस निर्देश को निष्पादित (Execute) कर देता है। अत: समान क्लॉक आवृत्ति (Clock Frequency) वाले अन्य माइक्रोकन्ट्रोलरों की अपेक्षा हार्वार्ड संरचना वाले माइक्रोकन्ट्रोलरों की चाल (speed) अधिक होती है।

## PIC माइक्रोकन्ट्रोलर्स के प्रकार (Types of PIC Microcontrollers)

8-बिट PIC माइक्रोकन्ट्रोलरों को उनकी आन्तरिक संरचना के आधार पर निम्नलिखित चार वर्गों में विभाजित कर सकते हैं—

- Base Line PIC
- Mid-range PIC
- Enhanced Mid-range PIC
- PIC 18x Series

### Base Line Core PIC माइक्रोकन्ट्रोलर

बेस लाइन PICs कम जटिल PIC माइक्रोकन्ट्रोलर है। ये 12-बिट निर्देश संरचना पर कार्य करते हैं जिसका अर्थ है कि इस प्रकार के माइक्रोकन्ट्रोलर के प्रत्येक निर्देश (Instruction) की "Word-size" 12-बिट होती है। इस प्रकार के PIC माइक्रोकन्ट्रोलर आकार में छोटे, हल्के एवं सस्ते होते हैं। ये 6 पिन से 40 पिन Packaging में उपलब्ध है। औद्योगिक क्षेत्र में इस प्रकार के PIC माइक्रोकन्ट्रोलर ने परम्परागत ICs जैसे कि—555ICs Logic Gates आदि का स्थान ले लिया है। बैक लाइन PIC माइक्रोकन्ट्रोलर निम्नलिखित श्रेणियों में उपलब्ध है—**12C5xx, 16C5xx** आदि; जैसे कि—12C508, 16C505 आदि।

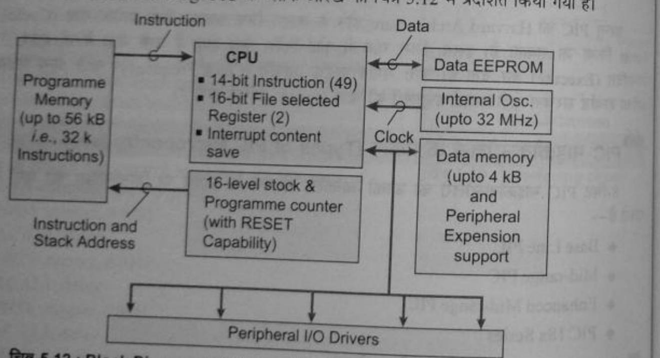
## Mid-range PIC माइक्रोकन्ट्रोलर

इस प्रकार के PIC माइक्रोकन्ट्रोलरों के प्रत्येक निर्देश की संरचना 14-बिट की होती है तथा ये PICs 20 MHz चाल तक कार्य कर सकती है। इस प्रकार के PICs 8 से 64 पिन पैकेज में उपलब्ध है। इस प्रकार के PIC माइक्रोकन्ट्रोलर निम्नलिखित बाहर से संयोजित की जा सकने वाली युक्तियों (Peripherals) के साथ उपलब्ध है—ADC, PWM (Pulse Width Modulation), Op-Amps एवं विभिन्न प्रकार के Protocols जैसे कि—USART Universal Synchronous/Asynchronous Receiver/Transmitter आदि। इसके कारण इस प्रकार के PICs, औद्योगिक क्षेत्रों के साथ-साथ अन्य कार्यों के लिए भी अति उपयोगी माइक्रोकन्ट्रोलर हैं।

मध्य-सीमा संरचना में PIC माइक्रोकन्ट्रोलर, निम्नलिखित नम्बरों के बाजार में उपलब्ध हैं—PIC 12 एवं PIC 16 द्वारा Labelled की गई है जैसे कि 16 cxx, 16 F 8 x, 16 F 87 x आदि। उदाहरण के लिए 16 C 62 A, 16 C 63, 16 C 64 A, 16 C 72, 16 C 73 A, 16 C 74 A, 16 F 84 आदि।

## वृद्धित मध्य-सीमा PIC माइक्रोकन्ट्रोलर (Enhanced mid-range PIC microcontroller)

माइक्रोचिप टेक्नोलोजी कं० ने वर्तमान की आवश्यकताओं के पूर्ण करने के साथ भविष्य के ग्राहकों को ध्यान में रखकर **"सुधारित हार्वाड संरचना (Modified Harvard Architecture)"** का प्रयोग कर 8-bit Enhanced Mid-range PIC माइक्रोकन्ट्रोलरों को निर्मित किया। न्यू निर्मित इन्हैन्स्ड मिड्-रैन्ज कोर में मिड-रैन्ज कोर के अच्छे घटकों को रखते हुए उसमे कुछ अन्य कार्यों के निष्पादन के लिए अन्य घटकों को सम्मिलित किया गया तथा निर्माण में यह भी ध्यान रखा गया है कि Mid-range PIC के स्थान पर Enhanced Mid-range PIC को भी सरलतापूर्वक उपयोग किया जा सके। Enhanced Mid-range PIC के ब्लॉक आरेख को चित्र 5.12 में प्रदर्शित किया गया है।




**चित्र 5.12 : Block Diagram of Enhanced Mid Range PIC Microcontroller (Using : Modified Hardware Architecture)**

मिड-रेन्ज PIC माइक्रोकन्ट्रोलरों की कुछ प्रमुख विशेषतायें (Features) निम्नलिखित है—

- कार्य-प्रदर्शन (Performance) में Mid-range PIC की अपेक्षा वृद्धि 50% तक
- कोर के आकार (Core-size) में कमी लगभग 40% तक
- 49 (14-Bit wide) सरल निर्देश
- 56 kB (*i.e.*, 32k Words Addresable) Flash Programme Memory



- 4 kB RAM/Data Memory
- 32 MHz तक का आन्तरिक दोलित्र (Internal Oscillator)
- 16-Level Hardware Stock
- 16-bit युक्त 2-file Select Registers
- ये 1.8 V से 5.5 V DC Low पावर में उचित प्रकार से कार्य कर सकते है।
- "C" Programming Language का उपयोग किया जा सकता है।
- ये Hardware Interrupt के साथ कार्य करने के अतिरिक्त इन्ट्रप्ट के Contents को स्टोर भी करता है।
- बहुश्रेणी-संचार (Multiple Serial Communications) एवं मोटर नियन्त्रण (Motor control) की योग्यता के अतिरिक्त अन्य कुछ Peripheral support निम्नलिखित है—
- Multiple Analog Digital Converters
- Multiple Comparators
- Multiple SPI (Synchronous Peripheral Interface), $I^2C$ (Inter IC connect) USART etc.
- Operational Amplifiers
- Non-volatile memory
- LCD Drive capability
- Multiple capture/compare/PWM
- m Touch TM sensing solution
- Support for future Peripheral Expansion

Enhanced Mid-range PIC माइक्रोकन्ट्रोलरों का नामकरण (नम्बर संख्या) निम्न प्रकार से किया गया है—12 F1xxx तथा 16 F 1 xx1.

## PIC 18 माइक्रोकन्ट्रोलर

सन् 2000 में Microchip Technology Co. ने 8-Bit के PIC 18 श्रेणी के माइक्रोकन्ट्रोलरों को निर्मित किया जोकि आज भी अधिकांश इम्बेडेड प्रणालियों में उपयोग किया जाते हैं। इसमें कुछ महत्वपूर्ण न्यू विशिष्टतायें (Features) निम्नलिखित है—

- 16-bit Instruction word
- A memory mapped Accumulator
- Read access to code memory (Table Reads)
- Direct Register to Register moves (Prior cores Needed to move Registers through Accumulator)
- An External Programme memory Interface to expand the code-space.
- An 8-bit × 8-bit Hardware Multiplier
- Auto-increment/decrement addressing controlled by control bits in a Status Register (ALUSTA)
- Call stock is 21 bits Wide and much deeper (31-level deep)
- The call-stock may be Read and written

- Conditional Branch Instructions
- Indexed Addressing Mode
- Extending the FSR (File Select Register) to 12-bits, allowing them to linearly address the Entire Data Address space.

PIC माइक्रोकन्ट्रोलरों का निम्न प्रकार से नामकरण (नम्बर संख्या) किया गया है—18 C 7 xx, 18 C 2xx, 18 Fxx, 18 Fxxx जैसे कि-18 F 452 PIC माइक्रोकन्ट्रोलर आदि।

## PIC माइक्रोकन्ट्रोलरों के गुण एवं दोष

### (1) गुण/लाभ (Merits/Advantages)

- PIC की संरचना में प्रयुक्त CPU, सरल तथा सस्ता (Cheapest) है।
- इनमें **हार्वार्ड संरचना (Harvard Architecture)** जिसमें निर्देश (Instructions) एक डाटा CPU में अलग स्रोत (source) से आते हैं, विद्यमान होने के कारण माइक्रो परिपथों को सरलता से अभिकल्पित (design) किया जा सकता है। इससे माइक्रोकन्ट्रोलर के मूल्य (cost) में एवं शक्ति उपयोग (power consumption) में कमी आती है तथा अधिक क्लॉक चाल (clock speed) प्राप्त की जा सकती है।
- **छोटा निर्देश समूह** (Small Instruction set)—इससे निर्देशों को कम समय में ही समझ सकते है तथा कम निर्देशों Flow-control को भी सरल बनाते हैं।
- **चयनित चाल एवं क्लॉक शुद्धता** (Selectable speed and Clock's Accuracy)—दोलित्र (Oscillator) का PIC की चिप पर ही निर्मित होने के कारण PIC एक निश्चित चयनित चाल पर ही कार्य करता है तथा कुछ समय पहले 12 F xxx आदि PICs में अति शुद्ध RC Oscillators का उपयोग किया गया है।
- **RISC Architecture**—इसमें Reduced Instruction set computer (RISC) संरचना का उपयोग किया गया है जिसमें प्रत्येक निर्देश एक clock चक्र में ही निष्पादित (execute) होता है जिससे माइक्रोकन्ट्रोलर की चाल (speed) में वृद्धि होती है।
- **बाहरी संयोजन की चौड़ी-सीमा** (Wide-range of Interfaces)—PIC के साथ बाहर की निम्नलिखित युक्तियों को संयोजित कर सकने की क्षमता होती है-ADCs (Analog-to-Digital Converters), USART (Universal Synchronous/ Asynchronous Receiver/Transmitter), PWM (Pulse Width Modulation), CAN (Controller Area Network), $I^2C$ (Inter IC Connect), Programmable Comparators, USB (Universal Serial Bus), Ethernet, SPI (SCSI Parallel, Interface; where, SCSI :Small Computer System Interface), LIN (Local Interconnect Network), PSP (Paint Shop Profile : Extension a popular Graphic Paint Programme) आदि।
- बहुत कम शक्ति खपत (Very-low Power consumption)
- **DIL (Dip-In-Line) Package :** PIC माइक्रोकन्ट्रोलर, DIL पैकेज में उपलब्ध है जिसका लाभ यह है कि अधिकांश उपभोक्ता को DIL पैकेज की ICs को हैन्डल (handle) करने की आदत होती है।
- सस्ते (Inexpensive) माइक्रोकन्ट्रोलर हैं।
- इनकी Programming एवं Debugging परिपथों में Energy Level सरल है।
- **स्थिर इन्ट्रप्ट अन्तर्हिति** (Interrupt Latency in Constant)—यदि हम PIC के साथ एक नियमित समय इन्ट्रप्ट (Regular Time Interrupt) को सैट करें तो हम पाते हैं कि सामान समय अन्तराल पर यह इन्ट्रप्ट प्राप्त होता रहता है जबकि AVR माइक्रोकन्ट्रोलर में कम से कम एक चक्र (One cycle) का समय क्षय (Jitter) होता है।



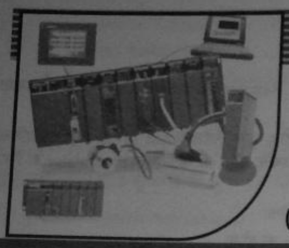
### (2) दोष/सीमायें (Demerits/Limitations)

- PIC में केवल एक Accumulator होता है।
- कई युक्तियों की Entire RAM को Access करने के लिए Register-bank Switching की आवश्यकता पड़ती है।
- PIC के रजिस्टर एवं प्रक्रियाओं का स्वतन्त्र (Orthogonal) होना-कुछ निर्देश RAM अथवा/और तात्कालिक स्थिरांक (Immediate Constants) को एड्रैस कर सकते हैं; जबकि अन्य का उपयोग Accumulator के लिए किया जा सकता है।
- निर्देशों के स्मृति सहायक (Mnemonic) नामों का स्वेच्छाचारी होना।
- अप्रत्यक्ष-कार्यवाही (Indirection) के लिए केवल एक Pointer का होना।
- Lookup तालिकाओं का अप्राप्त होना।
- Small Stock होने के कारण, Carry सहित Add एवं Subtract प्रक्रियायें सम्भव नहीं है।

## अभ्यासीय प्रश्न

1. एसेम्बली भाषा (Assembly Language) क्या है ? समझइए। तथा इसके लाभ व हानियाँ लिखिए।
2. MCS-51 के विभिन्न एसेम्बलर निर्देशकों (Assembler Directives) के नाम सहित उनके उपयोग का वर्णन कीजिए।
3. एसेम्बलर के लाभों को समझाइये।
4. MCS-51 का सॉफ्टवेयर अनुरूपक का वर्णन करो।
5. 8051 माइक्रो-कन्ट्रोलर के संग कीपेड/कीबोर्ड को संयोजित करने की विधि का वर्णन करो।
6. सप्त खण्ड प्रदर्शक का माइक्रो-कन्ट्रोलर को संयोजित करने की विधि का वर्णन करो।
7. ADC0801 एनेलॉग-से-डिजिटल परिवर्तक का 8051 माइक्रो-कन्ट्रोलर को संयोजित करने की विधि का वर्णन करो।
8. PIC माइक्रोकन्ट्रोलर का ब्लॉक आरेख देते हुए वर्णन करो।
9. PIC माइक्रोकन्ट्रोलर के प्रमतुख प्रकारों का वर्णन करो।
10. PIC माइक्रोकन्ट्रोलर के गुण एवं दोषों का वर्णन करो।

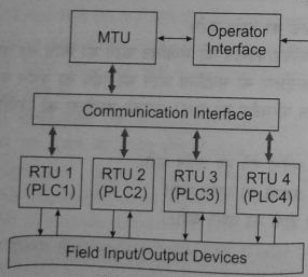
❑



# 6 सुपरवाइजरी कन्ट्रोल एवं डाटा एक्वीजीशन

## (Supervisory Control And Data Acquisition)

SCADA सुपरवाइजरी कन्ट्रोल एवं डाटा एक्वीजीशन का संक्षिप्त रूप है। SCADA एक तकनीक है जिसकी सहायता से किसी सिस्टम में नियमित इन्स्ट्रक्शन भेजे जाते हैं। SCADA सिस्टम की सहायता से किसी इन्डस्ट्रीज में विभिन्न प्रकार के ऑपरेशन परफॉर्म किये जाते हैं, जैसे—वॉल्व को ओपन तथा क्लोज करना, स्विच को ओपन तथा क्लोज करना, मॉनीटर को किसी प्रोसेस के लिए अलार्म करना आदि। SCADA एक सॉफ्टवेयर अधारित सिस्टम है, जिसकी प्रोसेस ऑनलाइन होती है। SCADA सिस्टम का उपयोग पावर ट्रांसमिशन, ऑयल तथा गैस ट्रांसमिशन, वाटर डिस्ट्रीब्यूशन तथा विभिन्न प्रकार की इन्डस्ट्रियल प्रोसेस में किया जाता है। SCADA सिस्टम में हार्डवेयर तथा सॉफ्टवेयर की सहायता से मानव तथा मशीन के बीच इन्टरफेस कराया जाता है। इस प्रकार किसी सिस्टम में वाटर टैंक को ओवरफ्लो से राकना है तो इसमें PLC तथा SCADA सिस्टम को उपयोग में लाया जाता है।

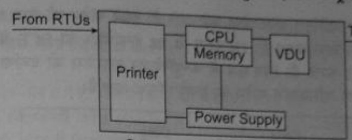



चित्र 6.1 : SCADA सिस्टम

SCADA सिस्टम को सर्वप्रथम सन् 1960 में एक मौसम विज्ञान विभाग द्वारा प्रयोग में लाया गया। इसमें विभिन्न प्रकार के मौसम से सम्बन्धित प्राप्त डाटाओं को कम्प्यूटर की सहायता से कम्पेयर किये गये तथा उन्हें एक विशेष उद्देश्य के रूप में प्रयोग किया गया। SCADA सिस्टम के ब्लॉक डायग्राम को चित्र 6.1 में दर्शाया गया है।

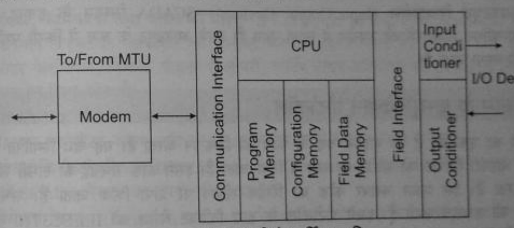
SCADA सिस्टम के ब्लॉक डायग्राम को निम्नलिखित भागों में बाँटा गया है।

**( अ ) मास्टर टर्मिनल यूनिट** (Master Terminal Unit)—यह यूनिट एक होस्ट कम्प्यूटर के नाम से जानी जाती है। MTU की सहायता से रिमोट टर्मिनल यूनिट को सूचना दी जाती है जो आपस में एक कम्युनिकेशन इन्टरफेस द्वारा एक-दूसरे से जुड़े रहते हैं। इस प्रकार RTU भी MTU को सूचना प्रदान करता है। MTU एक मास्टर यूनिट होती है जो कम्युनिकेशन द्वारा सभी सम्बन्धित प्रोसेसों को पूर्ण करती है। MTU के ब्लॉक डायग्राम को चित्र 6.2 में दर्शाया गया है।




चित्र 6.2 : मास्टर टर्मिनल यूनिट

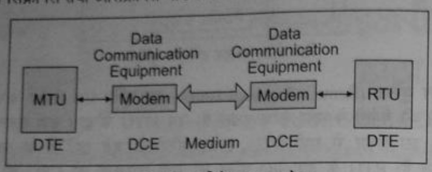
**( ब ) रिमोट टर्मिनल यूनिट** (Remote Terminal Unit)—यह यूनिट एक रिमोट कन्ट्रोल यूनिट होती है जो विभिन्न प्रकार की सूचनाओं को मैमोरी में स्टोर करके रखती है। जब MTU के द्वारा कुछ सूचना माँगी जाती है तब इसे RTU उपलब्ध कराता है। इस प्रकार से सभी RTU, PLC यूनिटों से जुड़े रहते हैं जो सभी प्रकार के कन्ट्रोलिंग ऑपरेशनों को परफॉर्म करते हैं। RTU के द्वारा I/O यूनिट की सभी सूचनाओं को CPU की सहायता से प्राप्त किया जाता है तथा उन्हें MTU को दे दिया जाता है। इसके सिग्नलों में RS-232 इन्टरफेसिंग सिस्टम का उपयोग किया जाता है। जब MTU इन्स्ट्रक्ट होती है तब RTU वॉल्व को ओपन तथा क्लोज करता है। यह स्विच ऑन या ऑफ की सहायता से एनालॉग सिग्नल सेट पॉइन्ट को रिप्रेजेन्ट करते हैं। इस प्रकार से इसकी पल्स ट्रेन स्टेपिंग मोटर की ओर घूमती है। RTU के ब्लॉक डायग्राम को चित्र 6.3 में दर्शाया गया है।




चित्र 7.3 : रिमोट टर्मिनल यूनिट

**( स ) कम्युनिकेशन इन्टरफेस** (Communication Interface)—कम्युनिकेशन इन्टरफेस MTU तथा RTU की CPU तथा अन्य डिवाइसों को इन्स्ट्रक्शन देने का माध्यम है। इसकी सहायता से किसी भी डाटा को मॉडेम द्वारा RTU से MTU में भेजा जाता है। इस इन्टरफेस की सहायता से किसी वॉल्व या स्विच को मानव तथा मशीन इन्टरफेस के द्वारा ऑन तथा ऑफ किया जाता है। इस इन्टरफेस में विभिन्न प्रकार के केबिलों तथा वायरों को उपयोग में लाया जाता है। इस प्रकार की इन्टरफेसिंग में असेम्बली लैंग्वेज का उपयोग किया जाता है जो हार्डवेयर तथा सॉफ्टवेयर को आपस में इन्टरफेस कराती है। SCADA सिस्टम में एक या एक से अधिक MTUs इन्स्ट्रक्शन को भेजने तथा डाटा को एक या एक से अधिक RTUs से प्राप्त करने के लिए कम्युनिकेशन लिंक की आवश्यकता होती है। जैसे-जैसे इसमें कुछ कम्युनिकेशन पाथ रिमोट साइड तथा सेन्ट्रल या मास्टर साइड स्थापित किये जाते हैं जिससे कि डाटा पास किया जा सके। सभी प्रकार के डाटा MTU तथा RTU के बीच बाइनरी डाटा मूव करता है। यदि यह स्टेटस कन्डीशन में है तब इसके स्विच ऑन-ऑफ होते हैं तथा यह बाइनरी फॉर्म से एनालॉग फॉर्म में कन्वर्ट होते हैं। इसमे बाइनरी के सिंगल स्ट्रिंग को एक के बाद दूसरे तक भेजा जाता है। इसमें सीरियल को पैरेलल मे परिवर्तित किया जाता है। इससे पैरेलल बसेस का उपयोग कम्प्यूटर के साथ किया जाता है तथा कम्प्यूटर से प्रिंटर के बीच भी किया जाता है लेकिन लम्बी दूरी की कम्युनिकेशन के लिए इसमें अतिरिक्त कीमत चुकानी पड़ती है। सीरियल डाटा ट्रांसफर मैथर्ड मे कुछ कन्वर्जन पहले MSB से LSB में होते हैं तथा कुछ कन्वर्जन LSB से MSB में होते हैं। यह सब कुछ कम्युनिकेशन प्रोटोकॉल पर निर्भर करता है।

प्रत्येक MTU तथा RTU डाटा, टर्मिनल इक्विपमेन्ट कहे जाते हैं। इसमें मॉडेम को डाटा कम्युनिकेशन इक्विपमेन्ट कहा जाता है जो DTE से इन्फॉर्मेशन प्राप्त करता है। इस प्रकार से यह इन्फॉर्मेशन को DCE में परिवर्तित करता है जो प्राप्त इन्फॉर्मेशन को DTE में ट्रांसफर करता है। चित्र 6.4 में कम्युनिकेशन इन्टरफेस को दर्शाया गया है। इस प्रकार के कम्युनिकेशन इन्टरफेस में सिंक्रोनस तथा असिंक्रोनस मॉडेम का उपयोग किया जाता है।




चित्र 6.4 : कम्युनिकेशन इन्टरफेस

**(द) ऑपरेटर इन्टरफेस** (Operator Interface)—ऑपरेटर इन्टरफेस एक मानव तथा मशीन इन्टरफेस है। इसकी सहायता से किसी इंजीनियर द्वारा किसी मशीन को कन्ट्रोल किया जाता है। इस प्रकार की इन्टरफेसिंग को ऑपरेटर इन्टरफेस कहते हैं।

**(इ) इनपुट/आउटपुट डिवाइसेस** (Input/Output Devices)—SCADA सिस्टम के इनपुट डिवाइस में कम्प्यूटर तथा उससे सम्बन्धित डिवाइसों को उपयोग में लाया जाता है। इसमें आउटपुट के क्रम में किसी मशीन या उससे सम्बन्धित ऑपरेशन को लाया जाता है।

## SCADA सिस्टम के कम्युनिकेशन प्रोटोकॉल

प्रोटोकॉल नियमों का एक सेट है जो बाइनरी वर्ड के पैटर्न को डिफाइन करता है। यह जब स्थापित होता है तब MTU से RTU की बाइनरी डिजिट को सीरीज के रूप में दिया जाता है। इसमें कोड सप्लाई की लम्बी सीरीज को 0 तथा 1 में उत्पन्न करता है। इस प्रकार बराबर कोड को रिसीव स्टेशन पर प्राप्त किया जाता है। सभी प्रकार के मैन्यूफैक्चर प्रोटोकॉल को डवलव करते हैं। इसमें प्रोटोकॉल के द्वारा निश्चित मैसेज को IEEEC3710 लेआउट की सहायता से भेजा जाता है। चित्र 6.5 में प्रोटोकॉल के लेआउट को दर्शाया गया है।

| Synch | Remote Address | Function | Internal Address | Modifier | Special Delivery Codes | Data | CRC |
|---|---|---|---|---|---|---|---|
| 8-Bits | 8-Bits | 8-Bits | 8-Bits | 8-Bits | 8-Bits | 0 to 192-Bits | 16-Bits |

चित्र 6.5 : प्रोटोकॉल ले-आउट

SCADA सिस्टम में उपयोग किये जाने वाले प्रमुख कम्युनिकेशन प्रोटोकॉल निम्नलिखित हैं—

**(i) सिंक** (Synch)—यह फ्रेम सिग्नल्स के द्वारा सभी पोटेन्शियल मैसेज को सिंक्रोनाइज क्लॉक तथा ट्रांसमीटर क्लॉक के द्वारा रिसीव तथा ट्रांसमिट किया जाता है।

**(ii) रिमोट एड्रेस** (Remote Address)—यह फ्रेम उस स्टेशन को डिफाइन करता है जहाँ मैसेज भेजना है।

**(iii) फंक्शन** (Function)—फंक्शन फ्रेम 256 विभिन्न प्रकार के मैसेजों को डिफाइन करता है तथा जिसमें वह एक को चुनता है।

**(iv) इन्टर्नल एड्रेस** (Internal Address)—इन्टर्नल फ्रेम रजिस्टर के सेट को रिसीविंग स्टेशन पर डिफाइन करता है जहाँ पर मैसेज डायरेक्ट प्राप्त किया जाता है।

**(v) मॉडीफायर** (Modifier)—यह इन्टर्नल एड्रेस को मॉडीफाइ करता है तथा यह डिफाइन करता है कि कितना डाटा इस मैसेज में जोड़ा गया है।



**(vi) स्पेशल डिलीवरी कोड** (Special Delivery Code)—यह शॉर्ट हैड मैसेज को RTU तथा MTU में रखता है।

**(vii) डाटा** (Data)—यह 0-192 फील्ड बिट्स की वेरियेबल लेन्थ है।

**(viii) साइक्लिक रिडन्डेन्सी कोड** (Cyclic Redundancy Code)—साइक्लिक रिडन्डेन्सी कोड का उपयोग ट्रांसमिशन एरर को डिटेक्ट करने में किया जाता है।

## SCADA के लाभ तथा उपयोग

### SCADA सिस्टम के उपयोग (Application of SCADA System)

SCADA सिस्टम के प्रमुख उपयोग निम्नलिखित हैं—

**(i) हाइड्रोलिक पावर स्टेशन** (Hydraulic Power Station)—SCADA सिस्टम का उपयोग विभिन्न हाइड्रोलिक पावर स्टेशन के ऑपरेशन्स को परफॉर्म करने में किया जाता है। इसके द्वारा टर्बाइन, जनरेटर, वाटरफ्लो, पावर ग्रिड ऑपरेशन मॉनीटर तथा कन्ट्रोल आदि फंक्शन परफॉर्म किये जाते हैं। इसमें सभी प्रकार के डाटा को लोकेशन को कन्ट्रोल रूम में प्राप्त तथा प्रेजेन्ट किया जाता है। पावर ग्रिड में विद्युत की आवश्यकता पर वाटरफ्लो को कन्ट्रोल करता है।

**(ii) ऑयल, कैमिकल्स तथा वाटर की पाइपलाइन** (Pipeline of Oil, Chemicals and Water)—SCADA सिस्टम का उपयोग पाइपलाइन ऑपरेशन को कन्ट्रोल करने में किया जाता है। स्टार्टिंग तथा स्टॉपिंग पम्प, लीकेज डायरेक्शन, मीटर केलीब्रेशन, डिमान्ड प्रोडक्ट डिलीवरी, शटिंग लाइन ऑपरेशन को SCADA सिस्टम की सहायता से मॉनीटर तथा कन्ट्रोल किया जाता है।

**(iii) ऑयल तथा गैस प्रोडक्शन** (Oil and Gas Production)—रिफाइनरी में SCADA सिस्टम का उपयोग किया जाता है। पम्प, फ्लोमीटर, वॉल्व आदि को SCADA सिस्टम से कन्ट्रोल किया जाता है।

**(iv) इलेक्ट्रीसिटी ट्रांसमिशन सिस्टम** (Electricity Transmission System)—इसमें लोड चेन्ज, स्विचेज, पावर बैलेन्स आदि में SCADA सिस्टम का उपयोग किया जाता है।

**(v) एरिगेशन सिस्टम** (Irrigation System)—केनाल गेट, ओपन या क्लोज वॉल्व, वाटर के फ्लो-रेट आदि में SCADA सिस्टम का उपयोग किया जाता है।

**(vi) मेट्रो ट्रेन** (Metro Train)—ट्रेक्शन ड्राइव, ब्रेकिंग तथा अन्य ऑपरेशनों में SCADA सिस्टम का उपयोग किया जाता है।

**(vii) एनीमेशन सिस्टम** (Animation System)—आज कल सभी प्रकार के एनीमेशन सिस्टम में SCADA सिस्टम का उपयोग किया जाता है।

### SCADA सिस्टम के लाभ (Advantages of SCADA System)

SCADA सिस्टम के प्रमुख लाभ निम्नलिखित हैं—

(i) इसके द्वारा इन्डस्ट्रीज में लगने वाली केप्टिल कॉस्ट को कम किया जाता है।
(ii) इसके द्वारा प्रोसेस ऑपरेशन की कॉस्ट को घटाया जाता है।
(iii) इसके द्वारा किसी सिस्टम की मेन्टीनेन्स कॉस्ट को कम किया जाता है।
(iv) इसकी सहायता से पावर को बचाया जाता है।
(v) इसकी सहायता से प्रोसेसिंग टाइम को कम किया जाता है।
(vi) इसके द्वारा ऑपरेशन में होने वाले फेल्योर को ऑन लाइन ठीक किया जाता है।
(vii) इसकी सहायता से डिफेक्ट प्रोडक्ट की संख्या को कम किया जाता है।

## इन्डस्ट्रियल ऑटोमेशन (Industrial Automation)

ऑटोमेशन शब्द का उपयोग सबसे पहले फोर्ड मोटर कम्पनी ने सन् 1940 में किया। ऑटोमेशन का प्रस्ताव सन् 1947 में रखा गया इसमें विभिन्न कार्यों को मशीनों तथा कन्ट्रोलिंग की सहायता से ऑपरेट तथा कन्ट्रोल किया जाता था। ऑटोमेशन का मतलब है कि बिना मानव किसी ऑपरेशन या मैन्यूफैक्चरिंग प्रक्रिया को सम्पन्न करना। ऑटोमेशन टेक्नोलॉजी का उपयोग मैकेनिकल, इलेक्ट्रॉनिक, इलेक्ट्रिकल तथा कम्प्यूटर आधारित सिस्टम में किया जाता है। इसकी सहायता से विभिन्न ऑपरेशनों तथा कन्ट्रोल प्रोडक्शन को ऑटोमेटिक सम्पन्न किया जाता है।

यह सत्य है कि वर्तमान युग ऑटोमेशन पर टिका हुआ है प्रत्येक नयी इन्डस्ट्रीज तथा पुरानी इन्डस्ट्रीज में ऑटोमेशन का उपयोग कर प्रोडक्शन दर बढ़ाने की होड़ मची हुई है। आधुनिक युग में जितनी भी नयी इन्डस्ट्रीज को खोला जा रहा है उन सभी में ऑटोमेशन सिस्टम/मॉर्डन प्लांट का उपयोग किया जा रहा है। प्रत्येक पुरानी इन्डस्ट्रीज ऑटोमेशन, मॉडराइजेशन का उपयोग कर पुरानी डिवाइसों को नयी डिवाइसों में परिवर्तित कर रही है। PLC एक इलेक्ट्रॉनिक डिवाइस है जिसका उपयोग दिन-प्रतिदिन इन्डस्ट्रियल ऑटोमेशन में बढ़ता जा रहा है। इस प्रकार के ऑटोमेशन से समय की बचत की जाती है तथा केपिटल कॉस्ट को बढ़ाया जाता है।

इस प्रकार के ऑटोमेशन का उद्देश्य किसी भी कार्य को ऑटोमेटिक तरीके से सम्पन्न करना तथा प्रत्येक कार्य को सेल्फ एक्टिंग, सेल्फ रेगुलेटिंग तथा सेल्फ रिलाइवेन्ट आदि से सम्पन्न करना है। इसमें प्रत्येक कार्य को प्रेक्टीकली सम्पन्न किया जाता है तथा इसमें निश्चित प्रोग्रामिंग कर उपयोग किया जाता है। ऑटोमेशन तकनीक के अन्तर्गत आने वाले प्रमुख बिन्दु निम्नलिखित हैं—

(i) प्रत्येक पार्टस के लिए ऑटोमेटिक मशीन टूल्स,
(ii) ऑटोमेटिक असेम्बली मशीन,
(iii) इन्डस्ट्रियल रोबोट्स,
(iv) ऑटोमेटिक मेटेरियल हैंडलिंग तथा स्टोरेज सिस्टम,
(v) क्वालिटी कन्ट्रोल के लिए ऑटोमेटिक इन्स्पेक्शन सिस्टम,
(vi) फीडबैक कन्ट्रोल तथा कम्प्यूटर प्रोसेस कन्ट्रोल,
(vii) प्लानिंग, डाटा कलेक्शन तथा सपोर्ट मैन्यूफैक्चरिंग एक्टीविटी के लिए कम्प्यूटर सिस्टम।

## ऑटोमेशन के लक्ष्य (Gaols of Automation)

(i) मैन्यूफैक्चरिंग ऑपरेशन के विभिन्न पहलुओं का समाकलन करना जिससे कि प्रोडक्ट क्वालिटी तथा एकरुपता, साइकिल टाइम तथा श्रमिक कॉस्ट को कम करना आदि प्रक्रियाएँ की जाती हैं।
(ii) अच्छे प्रोडक्शन की सहायता से मैन्यूफैक्चरिंग कॉस्ट को कम करना तथा प्रोडक्टविटी को सुधारना, पार्टस को लोडिड करना तथा मशीन को अनलोड करके प्रभावित बनाना आदि कार्य किया जाता है जिससे कि मशीन को प्रभावी रूप से उपयोग किया जाये।
(iii) अधिक से अधिक प्रोसेस दोहराकर क्वालिटी को बढ़ाना।
(iv) मानवीय हस्तक्षेप को कम करना एवं मानवीय गलतियाँ हटाना।
(v) मैनुअल हैंडलिंग के द्वारा डेमेज वर्कपीस को कम करना।
(vi) सेफ्टी के लेवल को बढ़ाना तथा खतरनाक स्थिति को घटाना।
(vii) मैन्यूफैक्चरिंग प्लांट पर फ्लोर स्पेश में कम खर्च करना तथा मशीन, मेटेरियल मूवमेन्ट तथा सम्बन्धित यंत्रों को प्रभावित बनाना।
(viii) अधिक से अधिक प्रशिक्षित श्रमिकों को रखना जिससे भविष्य में आने वाली तकनीकों को आसानी से समझाया जा सके।
(ix) बाजार में आने वाले प्रत्येक प्रोडक्ट के बारे में जानकारी रखना तथा नया प्रोडक्ट ग्राहकों को उपलब्ध कराना।



(x) नयी उत्पादन विधियों का उपयोग करके अधिक से अधिक लाभ कमाना।

## इन्डस्ट्रियल ऑटोमेशन के प्रकार (Types of Automation Industrial)

### (i) मैन्यूफैक्चरिंग ऑटोमेशन (Manufacturing Automation)

मैन्यूफैक्चरिंग ऑटोमेशन के अन्तर्गत आने वाले प्रमुख प्रकार निम्नलिखित हैं—

**(अ) फिक्स्ड ऑटोमेशन** (Fixed Automation)—फिक्सड ऑटोमेशन एक सिस्टम है जिसमें प्रोसेसिंग ऑपरेशन को इक्यूपमेन्ट कॉन्फिग्युरेशन की सहायता से फिक्स किया जाता है। इसमें ऑपरेशन का क्रम साधारणतया सरल होता है। यह समाकलित तथा कॉर्डीनेशन की सहायता से इस प्रकार के ऑपरेशनों को एक इक्यूपमेन्ट की सहायता से ऑपरेट करता है जो सिस्टम को कॉम्प्लेक्स बनाता है। फिक्सड ऑटोमेशन के लिए इक्नामिक संतुष्टी प्रोडक्ट से प्राप्त होती है जो उच्च कीमत तथा वॉल्यूम पर प्राप्त होती है। सन् 1924 में कन्ट्रोलरूम को मैकेनिज्ड कनवेयर्स, मैनुअल ऑपरेशन से परिचय कराया गया।

**(ब) प्रोग्रामेबल ऑटोमेशन** (Programmable Automation)—प्रोग्रामेबल ऑटोमेशन में प्रोडक्ट को इक्यूपमेन्ट की क्षमता के अनुसार डिजाइन किया जाता है। इसमें ऑपरेशन का क्रम विभिन्न प्रोडक्ट कान्फिग्यूरेशन के अनुसार अनुकूल बनाया जाता है। ऑपरेशन क्रम को प्रोग्राम द्वारा कन्ट्रोल किया जाता है जो इन्स्ट्रक्शन कोडेड को सेट करता है ताकि सिस्टम को पढ़ा तथा इन्ट्रप्ट किया जा सके। नये प्रोडक्ट को उत्पन्न करने के लिए, नये प्रोग्राम को तैयार तथा सिस्टम में प्रवेश कराना पड़ता है।

ऑटोमेटेड सिस्टम प्रोग्रामेबल है जिसका उपयोग लो वॉल्यूम प्रोडक्शन में किया जाता है। इसमें पार्ट या प्रोडक्ट बैच में बनाये जाते हैं। प्रत्येक बैच में नये प्रोडक्ट को उत्पन्न करने के लिए सिस्टम को मशीन इन्स्ट्रक्शन के साथ रीप्रोग्राम करना पड़ता है जो नये प्रोडक्ट के अनुरूप होता हैं इसमें मशीन का भौतिक सेटअप भी परिवर्तित करना जरूरी होता है। इसमें टूल लोडिड किये जाते हैं तथा मशीन टेबल पर यंत्रों को सजा कर रखना तथा प्रोग्राम को प्रवेश कराना आदि कार्य किया जाता है। सन् 1952 में प्रारम्भिक अनुप्रयोग को न्यूमेरिकल कन्ट्रोल मशीन तथा इन्डस्ट्रियल रोबोट के लिए परिचय कराया गया।

**(स) फ्लैक्सिबल ऑटोमेशन** (Flexible Automation)—फ्लैक्सिबल ऑटोमेशन प्रोग्रामेबल ऑटोमेशन का विस्तार है। फ्लैक्सिबल ऑटोमेशन को पिछले 15 या 20 सालों से उपयोग में लाया जा रहा है तथा इसके सिद्धान्त के अनुसार प्रोडक्ट का उत्पादन किया जा रहा है। फ्लैक्सिबल ऑटोमेशन सिस्टम के द्वारा विभिन्न प्रकार के प्रोडक्टो को एक निश्चित समय में उत्पन्न किया जाता है। इसमें एक प्रोडक्ट के बाद दूसरा प्रोडक्ट उत्पन्न करने में कम समय खर्च होता है। इसमें प्रोडक्शन टाइम सिस्टम की रीप्रोग्रामिंग तथा फिजिकल सेटअप करने पर लॉस्ट नहीं होता है। इस प्रकार से यह सिस्टम विभिन्न कॉम्बीनेशन तथा शैडयूल को बैच में इन्सर्ट करता है तथा प्रोडक्ट को उत्पन्न करता है।

इस सिद्धान्त को सन् 1960 में मशीन ऑपरेशन परफॉर्मेन्श के लिए लाया गया। इसका उपयोग लगातार प्रोडक्शन सिस्टम में किया जाता है। यह प्रोग्राम पार्ट में बिना प्रोडक्शन टाइम खर्च किये हुए परिवर्तित करता है तथा यह भौतिक सेटअप को परिवर्तित करके प्रोडक्शन टाइम को बचाता है।

### (ii) नॉन-मैन्यूफैक्चरिंग ऑटोमेशन (Non-Manufacturing Automation)

यह ऑफिस ऑटोमेशन तथा इन्ट्रीग्रेटेड डाटा प्रोसेसिंग मैकेनिज्म, ऑटोमेटिक एलीवेटर्स, ट्रांसपोर्टेशन, टिकिट सेलिंग इक्यूपमेन्ट आदि को रखता है। नॉन-मैन्यूफैक्चरिंग ऑटोमेशन को निम्नलिखित भागों में बाँटा गया है—

**(अ) ऑफिस ऑटोमेशन** (Office Automation)—यह इन्फॉर्मेशन को इनपुट की तरह जोड़ता है। यदि कार्य प्रगति पर है तब यह निश्चित ही फाइनल प्रोडक्ट को उत्पन्न करता है। इसमें इन्फॉर्मेशन फॉर्मेट, पेरोल रिप्रेजेन्टेशन, ट्रांसपोर्टेशन, रिर्जवेशन, शैडयूलिंग, बैंकिंग, सिक्योरिटी ट्रांजक्शन तथा कॉस्ट प्राइज एनालाइसिस आदि का उपयोग किया जाता है। यदि प्रोडक्टविटी बढ़ती है तब ऑफिस ऑटोमेशन के प्रत्येक श्रमिक को अधिक से अधिक लाभ प्राप्त होता है। इस प्रकार की इन्फॉर्मेशन की आवश्यकता बिजनिस मैनेजमेन्ट के लिए होती है जो इसे जल्दी से मैनेज कर लेते हैं।

इन्फॉर्मेशन को लम्बी दूरी तक ऑफिस ऑटोमेशन की सहायता से भेजा तथा इन्टीग्रेट किया जाता है।

**( ब ) होम ऑटोमेशन** (Home Automation)—यह इलेक्ट्रिकल डिवाइस/सिस्टम के ऑटोमेटिक ऑपरेशन को सम्मिलित करता है तथा उन्हें घरों में उपयोग किया जाता है।

**( स ) बिल्डिंग ऑटोमेशन** (Building Automation)—यह ऑटोमेटिक ऑपरेशन के लिए एसिस कन्ट्रोल सिस्टम, फायर हार्डज सिस्टम, इलेक्ट्रिकल लाइटिंग सिस्टम, पार्किंग सिस्टम, वाटर डिस्ट्रीब्यूशन सिस्टम तथा एयर कन्डीशनिंग सिस्टम आदि को सम्मिलित करता है। इस सिस्टम का उपयोग स्मार्ट बिल्डिंग में किया जाता है। इस सिस्टम को कन्ट्रोल करने के हिलए इन्टेलीजेन्ट सिस्टम/कम्प्यूटर का उपयोग किया जाता है।

## इन्डस्ट्रियल ऑटोमेशन के गुण (Features of Automation Industrial)

इन्डस्ट्रियल ऑटोमेशन के प्रमुख गुण निम्नलिखित है—

(i) इसकी प्रोडक्शन रेट उच्च होती है।

(ii) इसकी प्रारम्भिक लागत इंजीनियरिंग इक्यूपमेन्ट लगाने के लिए उच्च होती है।

(iii) प्रोडक्ट में परिवर्तन करने पर इसकी रिलायबिलिटी अच्छी नहीं होती है।

(iv) जनरल परपज इक्यूपमेन्ट में अधिक लागत लगती है।

(v) फिक्सड ऑटोमेशन के लिए प्रोडक्शन रेट कम तथा रिलायबल होती है।

(vi) प्रोडक्ट कॉन्फिग्युरेशन में फ्लैक्सिबिलिटी परिवर्तन के साथ जोड़ी जाती है।

(vii) यह सिस्टम बैच प्रोडक्शन के लिए अधिक अनुकूल है।

(viii) वेरियेबल मिक्चर के प्रोडक्ट के लिए प्रोडक्शन लगातार होता है।

(ix) इसमें मीडियम प्रोडक्शन भी सम्भव है।

(x) प्रोडक्ट डिजाइन वेरियेबल में फ्लैक्सिबिलिटी प्रोडक्ट के साथ सम्भव है।

## इन्डस्ट्रियल ऑटोमेशन के स्टेटजीज (strategies of Automation Industrial)

(i) **ऑपरेशन्य के लिए स्पेशलाइजेशन** (Combined Operations)—यह स्टेटजी स्पेशल परपज इक्यूपमेन्ट को डिजाइन करने में शामिल करती है तथा यह ऑपरेशन की उच्च क्षमता को परफॉर्म करती है।

(ii) **कम्बाइन्ड ऑपरेशन्स** (Specialization for Operations)—यह एक या एक से अधिक ऑपरेशन को मशीन से जोड़ता है। यह एक से अधिक मशीन की आवश्यकता को घटाता है।

(iii) **सायमनटेनियस ऑपरेशन्स** (Simultaneous Operations)—इसमें दो या दो से अधिक ऑपरेशन को एक साथ परफॉर्म किया जाता है तथा सम्पूर्ण प्रोसेस टाइम को घटाया जाता है।

(iv) **ऑपरेशन इन्टीग्रेशन** (Operation Integration)—यह स्टेटजी विभिनन वर्क स्टेशन को एक सिंगल इन्टीग्रेटेड मैकेनिज्म की सहायता से जोड़ती है। इस प्रोसेस में ऑटोमेटिड वर्क हैंडलिंग डिवाइस का उपयोग दो स्टेशनों के बीच पार्टों को ट्रांसफर करने में किया जाता है। इस प्रोसेस में विभिन्न पार्टों को एक वर्क स्टेशन से अन्य वर्क स्टेशन पर भेजा जाता है, जिसकी सहायता से सिस्टम के समस्त आउटपुट प्रोसेस को बढ़ाया जाता है।

(v) **फलैक्सिबिलिटी बढ़ाना** (Increased Flexibility)—यह स्टेटजी एक ही प्रोसेस में उत्पादों की विविधता के लिए उपयोग की जाती है। इसमें यंत्रों का अधिकतम उपयोग किया जाता है। इसमें प्राइम ऑब्जेक्ट प्रोडक्शन मशीन का सेटअप टाइम तथा प्रोग्रामिंग टाइम घटाया जाता है।

(vi) **मेटेरियल हैंडलिंग तथा स्टोरेज को बेहतर करना** (Improved Material Handling and Storage)—मेटेरियल हैंडलिंग तथा स्टोरेज सिस्टम में ऑटोमेशन शामिल रहता है। यह मैन्यूफैक्चरिंग टाइम को कम करता है।

(vii) **ऑन लाइन निरीक्षण** (On Line Inspection)—सामान्यत: निरीक्षण उसी समय सम्पन्न कर लिया



जाता है जब प्रोडक्ट को प्राडयूस किया जाता है। निरीक्षण के द्वारा इन्फॉर्मेशन प्रक्रिया में सुधार के लिए उत्पाद को परमिट किया जाता है। इसमें स्क्रैप तथा प्रोडक्ट के स्पेसीफिकेशन को डिजाइनर द्वारा सम्पन्न करता है।

(viii) **प्रोसेस कन्ट्रोल तथा ऑप्टीमाइजेशन** (Process Control and Optimization)—यह कन्ट्रोल स्कीम की वृहद श्रृंखला को सम्मिलित करता है तथा इसके द्वारा स्वयं की प्रोसेस तथा एसोसियेट इक्यूपमेन्ट आदि को प्रोसेसों को अधिक कुशलता से सम्पन्न करता है। इस स्टेटजी के द्वारा अलग-अलग प्रक्रियाओं को कम तथा अधिक उत्पाद की गुणवत्ता के लिए उपयोग में सुधार किया जाता है।

(ix) **प्लांट ऑपरेशन्य तथा कन्ट्रोल** (Plant Operations and Control)—यह स्टेटजी प्लांट लेवल पर कन्ट्रोल को शामिल करती है। यह प्रबन्धन तथा समन्वय लाने वाले प्लांट का सम्पूर्ण संचालन अधिक कुशलता से करने का प्रयास करता है। उसके अन्तर्गत प्लांट के अन्दर की नेटवर्किंग को शामिल किया जाता है।

## इन्डस्ट्रियल ऑटोमेशन के लाभ तथा हानियाँ (Advantages and Disadvantage of Industrial Automation)

### इन्डस्ट्रियल ऑटोमेशन के लाभ (Advantages of Industrial Automation)

इन्डस्ट्रियल ऑटोमेशन के प्रमुख लाभ निम्नलिखित हैं—

(i) **प्रोडक्टविटी बढ़ाना** (Increased Productivity)—ऑटोमेशन, श्रमिकों की अपेक्षा अधिक प्रोडक्टविटी को बढ़ाता है। इसकी सहायता से विभिन्न ऑटोमेशन फंक्शनों को परफॉर्म किया जाता है तथा प्रोडक्ट की दर को बढ़ाया जाता है।

(ii) **सेफ्टी** (Safety)—ऑटोमेशन के द्वारा ऑपरेशन तथा ट्रांसफरिंग के दौरान सेफ्टी प्रदान की जाती है। इसमें विभिन्न प्रकार के सेफ्टी यंत्रों का उपयोग किया जाता है। सेफ्टी सिस्टम की सहायता से प्रोसेस में होने वाले फैल्युअर को रोका जाता है।

(iii) **रॉ मेटेरियल की हाई कॉस्ट** (High Cost of Row Material)—इसमें रॉ मेटेरियल को एक निश्चित ऑपरेशन तथा प्रोडक्ट के लिए उपयोग किया जाता है जिससे प्रोडक्ट की कीमत को बढ़ाया जाता है। प्रोडक्ट बनने के बाद जो स्क्रैप बचता है उसे एक निश्चित ऑपरेशन की सहायता से रॉ मेटेरियल में बदला जाता है।

(iv) **प्रोडक्ट क्वालिटी को बढ़ाना** (Improved Product Quality)—ऑटोमेशन की सहायता से प्रोडक्ट की क्वालिटी को बढ़ाया जाता है। इसकी सहायता से केवल पार्ट को उत्पन्न नहीं किया जाता है बल्कि इसके मैनुअल काउन्टर पार्ट को भी सम्पन्न किया जाता है। यह विभिन्न पार्टों की क्वालिटी स्पेसीफिकेशन के साथ समानता तथा स्थिरता बनाये रखता है।

(v) **प्रोसेस इन्वेन्ट्री को कम करना** (Reduction of Process Inventory)—ऑटोमेशन की सहायता से इन्वेन्ट्री को एक निश्चित अनुपात में रखा जाता है। प्रोसेस इन्वेन्ट्री कॉस्ट पर निर्भर करता है जो कार्य की प्रोग्रेस को रिप्रेजेन्ट करता है। यह इन्डस्ट्रीज के कार्य में लगने वाले समय को मेनटेन करता है।

(vi) **मैन्यूफैक्चरिंग लीड टाइम में कमी** (Reduction in Manufacturing Lead Time)—ऑटोमेशन मैन्यूफैक्चरिंग प्रोडक्ट डिलीवरी तथा ग्राहक ऑर्डर के बीच में लगने वाले समय को कम करता है। यह ग्राहक को बेहतर सुविधा प्रदान करने के लिए मैन्यूफैक्चरर प्रतिस्पर्धा को बेहतर बनाना है तथा इसमें लगने वाले समय को कम करता है।

(vii) **श्रमिकों को घटाना** (Reduction of Labor)—ऑटोमेशन की सहायता से श्रमिकों को घटाया जाता है। इसमें सभी कार्यों को ऑटोमेटिक तरीके से सम्पन्न किया जाता है।

(viii) **सिस्टम को प्रोग्रामेबल बनाना** (Making the System Programmable)—ऑटोमेशन को प्रोग्राम की सहायता से सम्पन्न किया जाता है। इसमें प्रत्येक प्रोडक्ट को एक निश्चित प्रोग्राम के द्वारा ही प्रोडयूस

किया जाता है।

(ix) **सॉफ्टवेयर डवलपमेन्ट** (Software Development)—ऑटोमेशन के प्रत्येक ऑपरेशन को सॉफ्टवेयर प्रोग्राम की सहायता से सम्पन्न किया जाता है। इसमें विभिन्न इन्डस्ट्रियल सॉफ्टवेयर तथा कोड उपयोग किये जाते हैं जिससे प्रोडक्ट की सिक्योरिटी को बढ़ाया जाता है।

## इन्डस्ट्रियल ऑटोमेशन की हानियाँ (Disadvantages of Industrial Automation)

इन्डस्ट्रियल ऑटोमेशन की प्रमुख हानियाँ निम्नलिखित हैं—

(i) इसमें अधिक से अधिक केपिटल कॉस्ट खर्च होती है।

(ii) इसकी मैन्यूफैक्चरिंग फ्लैक्सिबिलिटी कम है।

(iii) प्रोडक्ट की माँग कम होने पर प्लांट को किसी अन्य उद्देश्य के लिए उपयोग नहीं किया जा सकता है।

(iv) ऑटोमेशन के प्रभाव से बेरोजगारी बढ़ती है।

(v) इसमें प्लांट में जहाँ पर ऑटोमेशन सिस्टम उपयोग किया जाता है यदि वहाँ पर किसी पार्ट में फैल्युअर होता है तो सम्पूर्ण सिस्टम को बंद रखा जाता हैं

(vi) इसके पार्टस आसानी से उपलब्ध नहीं होते हैं।

(vii) नया सॉफ्टवेयर प्रोग्राम डवलप करना अत्यधिक कठिन या महँगा होता है।

## इन्डस्ट्रियल ऑटोमेशन के सिस्टम (Systems used for Industrial Automation)

(i) **कम्प्यूटर एडिड डिजाइन** (Computer Aided Design)—यह एक कम्प्यूटर आधारित सिस्टम है। इसे CAD भी कहते हैं। यह ड्राइंग को रिप्रेजेन्टेशन तथा रिप्रोडक्शन करना, ड्राइंग के साथ संबद्ध जानकारी का विकास करना, डिजाइन को उत्पन्न करना आदि कार्य करता है। CAD द्वारा डिजाइन तथा जियोमैट्रिक मॉडलिंग, इंजीनियरिंग एनालायसिस, कम्प्यूटर काइनेटिक्स तथा ड्राफटिंग जैसे विभिन्न फंक्शनों को परफॉर्म करता है। CAD कम्प्यूटर ग्राफिक्स प्रणाली के उपयोग के साथ जुड़ा हुआ फंक्शन है।

(ii) **कम्प्यूटर एडिड मैन्यूफैक्चरिंग** (Computer Aided Manufacturing)—यह एक कम्प्यूटर आधारित प्रणाली है जिसका उपयोग मैन्यूफैक्चरिंग प्रोसेस में किया जाता है। इसका उपयोग मशीन कन्ट्रोल ऑपरेशन्स, प्रोसेस प्लानिंग, फैक्ट्री मैनेजमेन्ट आदि में किया जाता है। इसे CAM भी कहते हैं।

(iii) **कम्प्यूटर एडिड इंजीनियरिंग** (Computer Aided Enginneering)—यह सिस्टम इंजीनियरिंग असिसटेन्स को CAD सिस्टम की सहायता से सहयोग प्रदान करता है। यह मैथमैटिकल मॉडलिंग, इंजीनियरिंग एलीमेन्ट एनालाइसिस आदि से सपोंट करता है। इसे CAE भी कहा जाता है।

(iv) **कम्प्यूटर इन्टीग्रेटेड मैन्यूफैक्चरिंग** (Computer Integrrated Manufacturing)—यह सिस्टम स्वाचालन कम्प्यूटर के उपयोग से मैन्यूफैक्चरिंग तथा उद्यम को एकीकरण प्रदान करता है। इस सिस्टम में डिजाइन, इन्वेन्ट्री कन्ट्रोल, फिकिजकल डिस्ट्रीब्यूशन, कॉस्ट एकाउन्टिंग, प्लानिंग, प्रोडक्शन कन्ट्रोल तथा खरीददारी आदि फंक्शन परफॉर्म किये जाते हैं। इस फंक्शन को डायरेक्ट मेटेरियल मैनेजमेन्ट, शॉप फ्लोर डाटा एक्वीजीशन तथा कन्ट्रोल के साथ सम्मिलित किया जाता है।

(v) **फ्लैक्सिबल मैन्यूफैक्चरिंग सिस्टम** (Flexible Manufacturing System)—यह कम्प्यूटर आधारित सिस्टम है जिसका उपयोग मशीन ऑपरेशन, मेटेरियल ट्रांसपोर्ट उद्देश्यों में किया जाता है। मशीन टूल के समूह को इस सिस्टम द्वारा संचालित किया जाता है। प्रोडक्शन डाउन टाइम, मशीन उपयोगिता आदि को रिकॉर्ड तथा रिपोर्ट की आवश्यकतानुसार तैयार किया जाता है।

(vi) **प्रोग्रामेबल कन्ट्रोलर्स** (Programmable Controllers)—इस प्रकार के सिस्टम में इन्डस्ट्रियल कम्प्यूटर का उपयोग प्रोसेस/मशीनिंग ऑपिरशन प्रक्रिया को कन्ट्रोल करने में किया जाता है। इसमें PLC डिजिटल/एनालॉग इनपुट/आउटपुट के द्वारा अर्थमैटिक लॉजिक, टाइमिंग, काउन्टिंग, सिक्वेंसिंग



तथा कन्ट्रोल ऑपरेशन को परफॉर्म करता है।

(vii) **कम्प्यूटर आधारित कन्ट्रोल सिस्टम, मेजरमेन्ट सिस्टम** (Computer Based Control System/ Measurement System)—इस प्रक्रिया में टेम्प्रेचर, प्रेशर, लेवल तथा फ्लो को ऑटोमेटिक मेजर तथा कन्ट्रोल किया जाता है। इन वेरियेबल का उपयोग प्रोसेस यूनिट, जैसे—बॉयलर, केमिकल रियेक्टर, हीट एक्सचेंजर, ड्रायर क्रिस्टलर आदि में किया जाता है। इसमें डिजिटल कम्प्यूटर का उपयोग कन्ट्रोलिंग डिवाइस के रूप में किया जाता है।

(viii) **रोबोट्स** (Robots)—इन्डस्ट्रियल रोबोट रिप्रोग्रामेबल, मल्टीफंक्शन में मैनिपुलेटर डिजाइन मेटेरियल को मूव करने के लिए, पार्ट आदि के लिए डिजाइन किये जाते हैं। इसमें स्पेशन डिवाइस का उपयोग विभिन्न प्रकार के मोशन कन्ट्रोल तथा अन्य प्रकार के फंक्शनों में किया जाता है। रोबोट में डिसीजन लेने की क्षमता, इनपुट को सेन्स तथा रिस्पॉन्ड करने की क्षमता तथा अन्य डिवाइसों तथा मशीनों से कम्युनिकेट करने की क्षमता होती है। रोबोट का उपयोग स्पोट बैल्डिंग, मेटेरियल ट्रांसफर, मशीन लोडिंग, स्प्रे पेन्टिंग तथा असेम्बली आदि में किया जाता है।

(ix) **SCADA**—यह सुपरवाइजरी कन्ट्रोल एवं डाटा एक्वीजीशन का संक्षिप्त रूप है। यह एक कन्ट्रोलिंग डिवाइस है जिसकी सहायता से इन्डस्ट्रीज में विभिन्न प्रकार के कन्ट्रोलिंग ऑपरेशन परफॉर्म किये जाते हैं। SCADA टेक्नोलॉजी उपयोगकर्ता को एक या एक से अधिक स्थानों से डाटा एकत्र तथा भेजने की सुविधा उपलब्ध कराती है। इसकी सहायता से उचित कन्ट्रोल इन्स्ट्रक्शन अन्य डिवाइसों को दिये जाते हैं। SCADA ऑपरेटर को आवश्यक रिमोट लोकशन से सम्बन्धित ऑपरेशन परफॉर्म करने की अनुमति देता है। इसमें सभी प्रकार के ऑपरेशन कन्ट्रोल रूप से कन्ट्रोल किये जाते हैं। SCADA सिस्टम ऑपरेटर को सेट पॉइन्ट बनाने तथा परिवर्तित करने की अनुमति देता है जिसकी सहायता से ऑपरेटर ओपन या क्लोज वॉल्व, स्विच, मॉनीटर अलार्म तथा गैस, ऑयल से डाटा को प्राप्त करता है एवं पाइपलाइन की प्रक्रिया को सम्पन्न करता है। अगर प्रोसेस ऑपरेशन की दूरी 100 या 1000 किलोमीटर है तब SCADA एक ओर से दूसरी ओर विभिन्न सूचनाओं का आदान-प्रदान कर सकता है। यही इसका मुख्य लाभ है।

(x) **डिस्ट्रीब्यूटिड कन्ट्रोल सिस्टम** (Distributed Control System)—डिस्ट्रीब्यूटिड कन्ट्रोल सिस्टम कम्प्यूटर का नेटवर्क है जिसका उपयोग प्रोसेस इन्डस्ट्रीज में विभिन्न प्रकार की ऑन लाइन प्रक्रियाओं को प्रदर्शित करने में किया जाता है। इसकी सहायता से किसी प्रोसेस को वीडियो की सहायता से मॉनीटर पर देखा जा सकता है। डिजिटल सिग्नल को एक स्थान से दूसरे स्थान तक भेजने के लिए सिंगल केबिल कम्युनिकेशन नेटवर्क का उपयोग किया जाता है जिससे वायरिंग की कीमत तथा जटिलता को घटाया जाता है। डिस्ट्रीब्यूटिड प्रोसेस आर्किटेक्चर विभिन्न प्रोसेसरों को फंक्शनल डिस्ट्रीब्यूशन टॉस्क तथा फैल्युअर की रिस्क को कम करने की अनुमति देता है। डिस्ट्रीब्यूटिड कन्ट्रोल तथा मॉनीटरिंग के द्वारा कम से कम वायरिंग का उपयोग करके विभिन्न कन्ट्रोलिंग प्रक्रियाओं को कन्ट्रोल रूम से जोड़ा जाता है। डिस्ट्रीब्यूटिड कन्ट्रोल सिस्टम की सहायता से लगभग 150 लूप कप्ट्रोल किये जाते हैं।

(xi) **कम्प्यूटर न्यूमेरिकल कन्ट्रोल** (Computer Numerical Control)—कम्प्यूटर न्यूमेरिकल कन्ट्रोल प्रोग्रोमेबल ऑटोमेशन के फॉर्म में होता है। इसमें प्रोसेस इक्यूपमेन्ट नम्बर, लेटर्स तथा अन्य सिम्बल्स की सहायता से कन्ट्रोल किये जाते हैं। इसमें नम्बर्स, लेटर्स तथा सिम्बल्स एक निश्चित फॉर्मेट में कोडिड रहते है जो एक निश्चित कार्यक्षेत्र या कार्यस्थिति को इन्स्ट्रक्शन प्रोग्राम की सहायता से डिफाइन करते हैं। इस सिस्टम में माइक्रो कम्प्यूटर एक मशीन कन्ट्रोल यूनिट के रूप में उपस्थित रहता है। जब किसी प्रोसेस कार्य को बदला जाता है तब इसका प्रोग्राम भी बदल दिया जाता है। इसमें मशीन टूल एप्लीकेशन, जैसे—ड्रिलिंग, मिलिंग, टर्निंग तथा ड्रॉफटिंग, इन्स्पैक्शन आदि फंक्शन उपयोग किये जाते हैं।

## इन्डस्ट्रियल ऑटोमेशन के अनुप्रयोग (Application of Automation Industrial)

(i) प्रोसेस इन्डस्ट्रीज में विभिन्न पैरामीटर, जैसे—टेम्प्रेचर, प्रेशर, लेवर, फ्लो आदि को ऑटोमेटिक सिस्टम से मॉनीटर तथा कन्ट्रोल किया जाता है। इसकी सहायता से बोतल, बैग, कार्टून तथा ड्रम फिलिंग आदि ऑपरेशन एक निश्चित सिक्वेन्स से कन्ट्रोल किये जाते हैं।

(ii) मैकेनिकल इन्डस्ट्रीज में, ड्रिलिंग, फिनिशिंग, कटिंग आदि कार्य वर्कशॉप में ऑटोमेटिक सिस्टम की सहायता से ऑटोमेटिकली परफॉर्म किये जाते हैं। इस सभी प्रक्रियाओं को CNC तथा FMS मशीन द्वारा ऑपरेट किया जाता है। इस प्रकार के सिस्टम की सहायता से वाटर डिस्ट्रीब्यूशन, गैस फिलिंग टर्मिनल्स, ऑयल फिलिंग टर्मिनल्स, इलेक्ट्रीसिटी जनरेशन, इलेक्ट्रीसिटी डिस्ट्रीब्यूशन, रिफाइनरीज आदि में ऑटोमेटिक SCADA सिस्टम का उपयोग किया जाता है।

(iii) आज-कल चाय, कॉफी की मशीन, माइक्रोवेव ओवन, वॉशिंग मशीन आदि में ऑटोमेटिक सिस्टम तथा PLC का उपयोग किया जा रहा है।

(iv) कार्गोशिप के विभिन्न ऑपरेशन्स को ऑटोमेटिक सिस्टम की सहायता से कन्ट्रोल किया जाता है।

(v) केमीकल रियेक्टर, बॉयलर, हीट एक्सचेन्जर, डॉयर, क्रिस्टलाइजर जैसी विभिन्न प्रक्रियाओं को ऑटोमेटिक सिस्टम की सहायता से मॉनीटर तथा कन्ट्रोल किया जाता है।

(vi) ऑटोमेशन सिस्टम का उपयोग कृषि यंत्रों, केमीकल डिवाइसों, पेट्रोलियम, सीमेन्ट इन्डस्ट्रीज आदि में किया जाता है। इन सभी सिस्टमों में DCS सिस्टम का उपयोग किया जाता है।

(vii) ऑटोमोबाइल इन्डस्ट्रीज, मैकेनिकल इन्डस्ट्रीज, स्पेश व्हीकल में ऑटोमेटिक रोबोटिक्स सिस्टम का उपयोग किया जाता है।

(viii) ऑटोमेटिक सिस्टम का उपयोग इलेक्ट्रॉनिक खिलौने, फ्रीजर, वाटर प्यूरीफाई सिस्टम, मॉडलिंग तथा सिमुलेशन आदि सिस्टमों में किया जाता है।

## मशीनीकरण तथा ऑटोमेशन में अन्तर (Difference between Mechanization and Automation)

| क्र०सं० | मशीनीकरण | ऑटोमेशन |
|---|---|---|
| (i) | इसमें सभी कार्य मानवीय शक्ति द्वारा सम्पन्न किये जाते हैं। | इसमें सभी कार्य ऑटोमेटिकली सम्पन्न किये जाते हैं। |
| (ii) | इसमें मानवीय शक्ति का उपयोग किया जाता है। | इसमें मानवीय मस्तिष्क का उपयोग किया जाता है। |
| (iii) | इसमें किसी भी प्रोसेस को एक निश्चित क्रम में नहीं किया जाता है। | इसमें किसी भी प्रोसेस को एक निश्चित क्रम में किया जाता है। |
| (iv) | यह बहुत पुराना पैटर्न है। | यह एक बिल्कुल नया पैटर्न है। |
| (v) | इसमें नेटवर्क सिस्टम की आवश्यकता कम रहती है। | इसमें नेटवर्क सिस्टम की आवश्यकता अधिक रहती है। |
| (vi) | इसमें अत्यधिक समय खर्च होता है। | इसमेंकम से कम समय खर्च होता है। |
| (vii) | किसी प्रोसेस को एक निश्चित समय में सम्पन्न नहीं किया जा सकता है। | किसी प्रोसेस को एक निश्चित समय में सम्पन्न किया जाता है। |
| (viii) | इसमें अधिक ऑपरेटरों की आवश्यकता होती है। | इसमें कम ऑपरेटरों की आवश्यकता होती है। |
| (ix) | इसमें गलती होने की सम्भावना अधिक रहती है। | इसमें गलती होने की सम्भावना कम रहती है। |
| (x) | इसमें फैल्युअर होने पर एक निश्चित पार्ट प्रभावित होता है। | इसमें फैल्युअर होने पर सम्पूर्ण सिस्टम प्रभावित होता है। |



## ऑटोमेशन की उपयोगिता (Utility of Automation)

(i) ऑटोमेशन सिस्टम की उपयोगिता पावर सिस्टम के विभिन्न ऑपरेशनों, जैसे—ट्रांसमिशन, डिस्ट्रीब्यूशन, जनरेशन आदि पर निर्भर करती है।

(ii) ऑटोमेशन की उपयोगिता प्रोसेस लॉजिक पर निर्भर करती है।

(iii) इससे प्लांट की दक्षता तथा प्राडक्टविटी को बढ़ाया जाता है।

(iv) यह प्रोसेस स्टेटस की जानकारी उपलब्ध कराती है।

(v) ऑटोमेशन की उपयोगिता मैकेनिकल सिस्टम, जैसे—टेम्प्रेचर, प्रेशर मॉनीटरिंग तथा अन्य प्रोसेसों पर निर्भर करती है।

(vi) इसकी सहायता से कन्ट्रोल उपकरणों को ऑटोमेशन के अनुकूल बनाया जाता है।

(vii) SCADA सिस्टम के विभिन्न ऑपरेशनों में ऑटोमेशन की उपयोगिता के विभिन्न प्रोसेसों को सम्पन्न किया जाता है।

## अभ्यासीय प्रश्न

1. SCADA सिस्टम क्या होता है?
2. SCADA सिस्टम को ब्लॉक डायग्राम की सहायता से समझाइये।
3. SCADA सिस्टम में कौन-कौन से कम्युनिकेशन प्रोटोकॉल उपयोग किये जाते हैं? समझाइये।
4. SCADA सिस्टम के उपयोगों को लिखिए।
5. SCADA सिस्टम के लाभों को लिखिए।
6. इन्डस्ट्रियल ऑटोमेशन क्या होता है?
7. ऑटोमेशन के लक्ष्यों को समझाइये।
8. इन्डस्ट्रियल ऑटोमेशन के प्रकारों को समझाइये।
9. इन्डस्ट्रियल ऑटोमेशन के लाभ तथा हानियों को समझाइये।
10. इन्डस्ट्रियल ऑटोमेशन के अनुप्रयोगों को समझाइये।

❑