Specifying Indirect Address by Using Index Register
By Chris Chiang (4/8/2008)

All of the PV models provide the index registers ($I) to support the indirect addressing. The index registers are battery backed if the HMI unit has battery backed memory. The index registers is cleared to zero by the operating system only when the panel application is updated. With the support of indirect addressing, an object or macro can be designed to access different sets of data at different time.

Example

1) The word address W[$I30] is equivalent to W2000 when the value of $I30 is 2000.
2) The word address $U[$I0+123] is equivalent to $U223 when the value of $I0 is 100.
3) The bit address $U[$I2].a is equivalent to $U0.a when the value of $I2 is 0.
4) The word address [$I2]:W100 is equivalent to 3:W100 when the value of $I2 is 3.
5) The bit address [$I0]:W[$I1+10].f is equivalent to 5:W20.f when the values of $I0 and $I1 are 5 and 10 respectively.

Note to software users

1) It is the user's responsibility to make sure that the values in the index registers will result in valid addresses at run-time. The software has no way to check the validity of the using of index registers.
2) The offset values must be a positive number and the maximum offset value is 4095.
3) Only $I0~$I15 can be used for the node address (PLC address) and no offset value is allowed.
4) Make sure the PLC driver you are going to use supports the indirect addressing.

Note to driver developers

See the following sample code to know how to calculate the real address of an I/O command.

```
struct IoCmnd_PV *pCmnd;
struct IoCmndHdr_PV *hdr = pCmnd->hdr;
DWORD addr;
…
addr = hdr->addr;
if (addr & 0x80000000) {// [31]: 1 - Indirect address
                        // [30..16]: index register no.
                        // [15..0]: offset
    addr = *(linkPar->pIndxReg + ((addr & 0x7fff0000) >> 16)) + (addr & 0x0000ffff);
}
```