EMERSON
Industrial Automation

Application Note CTSL #004

The Application Note is pertinent to the Unidrive SP, Commander GP20
and Commander SK using SyPT Lite

Decoding the I/O Status Byte

Introduction: Parameter #8.20 in the Unidrive SP and Commander SK provides a status byte that reveals the status of the terminal strip inputs and outputs all within 1 byte. This can be a convenient register to pass via communications to a PLC or HMI to efficiently convey the information on the status of the I/O versus addressing each I/O point individually. Another purpose of this data byte would be allow one to observe the status of the I/O visually however there is one problem- this data byte is expressed in decimal. For example if the UnidriveSP Terminals T24,T25 and T26 were active, #8.20 would read 7. If the same terminals were active and the Relay output active, #8.20 would read 71 (64+7).

This byte is used to determine the status of the digital I/O by reading one parameter.
The bits in this byte reflect the state of Terminal Strip I/O

			For CommSK	For Unidrive SP
	Bit	**Binary**	**Digital I/O**	**Digital I/O**
Lo Nibble	0	1	B3 input/output	T24 input / output 1
	1	2	B4 Input	T25 input / output 2
	2	4	B5 Input	T26 input / output 3
	3	8	B6 Input	T27 input 4
Hi Nibble	4	16	B7 Input	T28 input 5
	5	32		T29 input 6
	6	64	Relay Output	Relay Output
	7	128		T22 24V output

Unless one was well versed in Decimal to Binary conversion, the data byte in #8.20 is not as useful as a straight binary representation of the corresponding Terminal pin status.

For instance, if one were to look at a UnidriveSP register representing the LoNibble of this byte and see:

0111

They would come to the conclusion that Terminals T24,T25 and T26 were active

and if one were to look at a register representing the HiNibble of this byte and see 100, then one would conclude that the Relay Output is also active.

This would be the same if #8.20 read 71 or (64+7) $\underbrace{\mathbf{0100}}_{\text{Hi Nibble}}\ \underbrace{\mathbf{0111}}_{\text{Lo Nibble}}$

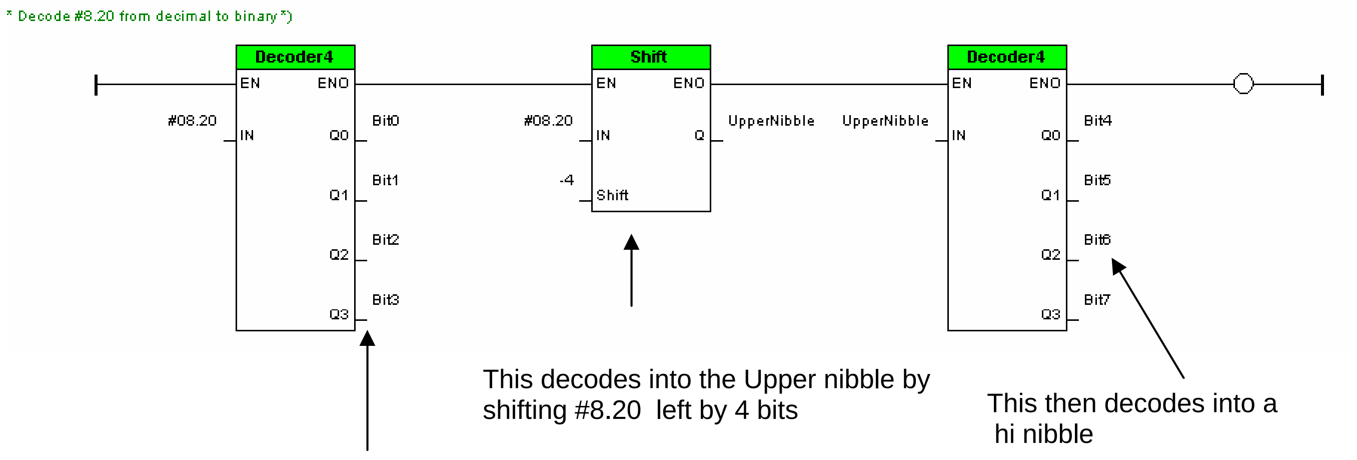
Objective:

This application note will illustrate a method to decode the decimal representation of #8.20 into 2 binary nibbles (collection of 4 bits – a Hi and Lo Nibble) which is inherently easier to visually interpret.

Implementation

SyPT Lite has several useful function blocks to assist with this conversion. Firstly there is a **Decimal to Binary decoder** and an adjustable **Shift Register**. Both will prove key to this implementation.

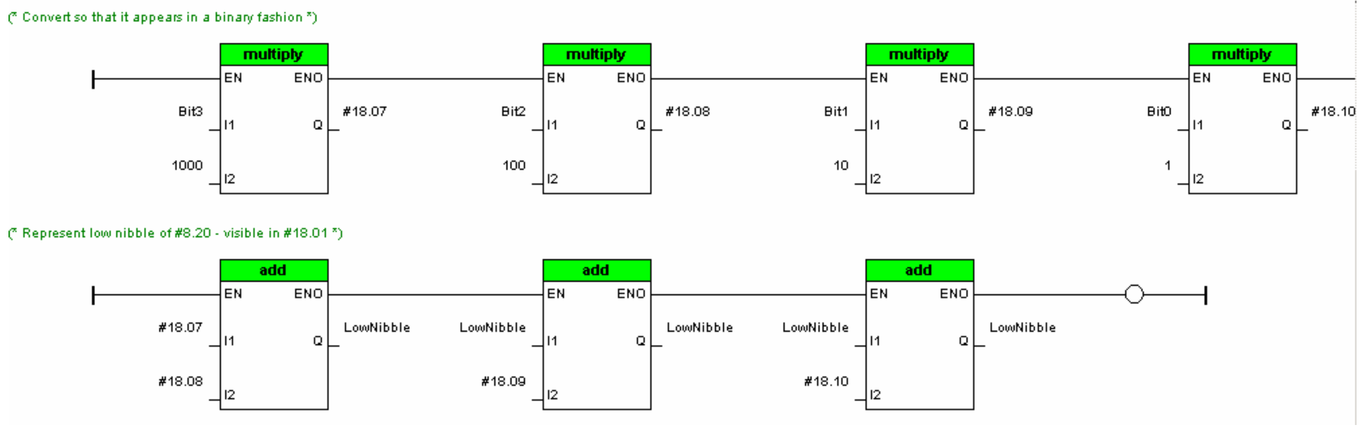
The first step that I took was to decode the status register which consists of a decimal number.



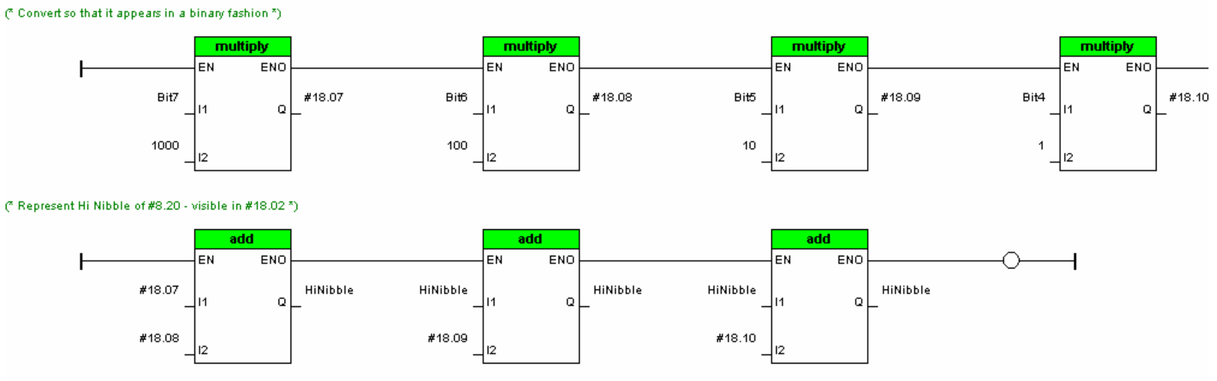
* Decode #8.20 from decimal to binary *)
Decoder4
EN ENO
#08.20
IN Q0 Bit0
Q1 Bit1
Q2 Bit2
Q3 Bit3
Shift
#08.20 IN Q UpperNibble
-4 Shift
Decoder4
UpperNibble IN Q0 Bit4
Q1 Bit5
Q2 Bit6
Q3 Bit7
This decodes into the Upper nibble by shifting #8.20 left by 4 bits
This then decodes into a hi nibble

This decodes into a lo nibble

Now what one can do is synthetically create a decimal number that are powers of 10 based on the bit position of the nibble. I decided to place each of these weighted values in separate registers (#18.07-18.10) then sum them all up into a register I called LowNibble.



Then I did the exact same thing for the more significant bits to become the HiNibble.



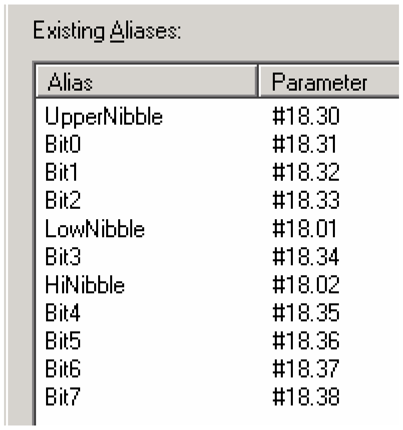
Register Assignments

Status Word expressed in binary can be observed here

LoNibble	#18.01
HiNibble	#18.02

Bit0	#18.31
Bit1	#18.32
Bit2	#18.33
Bit3	#18.34
Bit4	#18.35
Bit5	#18.36
Bit6	#18.37
Bit7	#18.38

#18.07 thru #18.10
temporary


Existing Aliases:

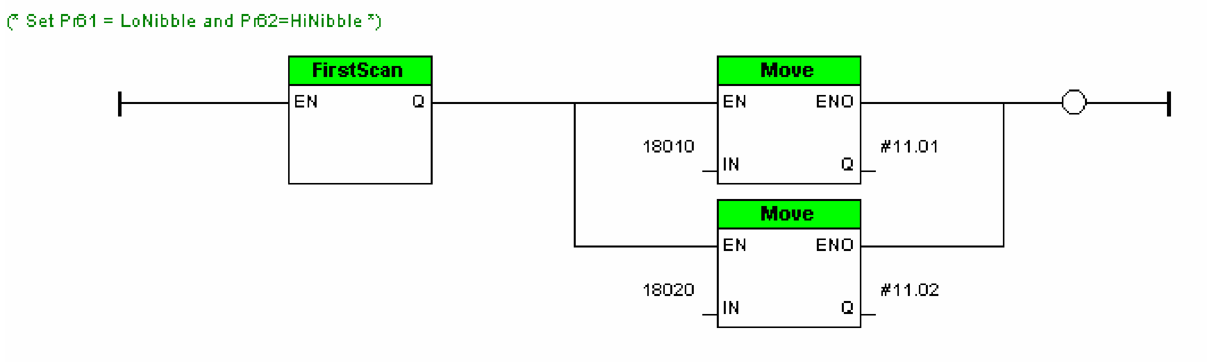
Alias	Parameter
UpperNibble	#18.30
Bit0	#18.31
Bit1	#18.32
Bit2	#18.33
LowNibble	#18.01
Bit3	#18.34
HiNibble	#18.02
Bit4	#18.35
Bit5	#18.36
Bit6	#18.37
Bit7	#18.38

In order to observe the Lo and Hi nibble, from the Commander SK on would need to set
Pr71= 18.01
Pr72= 18.02

And observe the Lo and Hi Nibble at Pr 61 and 62 respectively

Embellishment

However, we can do this programmatically as well using the rung of ladder as shown below:



This assignment obviously does not need to be done in the program but if there is room and nothing else needs done in the ladder, it makes setup automatic.

The above rung simply moves 1801 into #11.01 which is the same as Pr71
and moves 1802 into #11.02 which is the same as Pr72.

Summary

From this SyPT Lite example, one should realize that if the base drive does not have a certain desired function, often you could create your own functions required using the built-in PLC ladder functionality provided by SyPT Lite.

To obtain SyPT Lite click here → **SyPT Lite** Free

To obtain this code fragment click on the following link → **Status Byte Decoder**
(for Commander SK)

Questions ?? Ask the Author:

Author: **Ray McGranor** (716)-774-0093 **e-mail :** ray.mcgranor@emersonct.com