List of content

Introduction

A common way of sending instructions to the SyncPos option and to receive status information from the SyncPos option is by means of the digital inputs and outputs. This approach works fine in applications where the necessary communication can be transmitted as a limited number of signals (START, STOP, ALARM etc.)

However in many applications there is a need to transfer process data such as the real-time position of a lift system or the real-time position deviance of a synchronisation application.

To relay this kind of information a Fieldbus option can be used. Using a Fieldbus option can also help to free up the digital inputs and outputs for other tasks.

This note will provide the reader with a series of examples on how to use this flexible interface between a SyncPos option and an external control device (typically a PLC).

Working with PCD's

Communication with the SyncPos option is done by exchanging a number of PCD's or "process data words" (a two-byte integer value).
The number of PCD's range from 1 to 8 depending on the Fieldbus type:

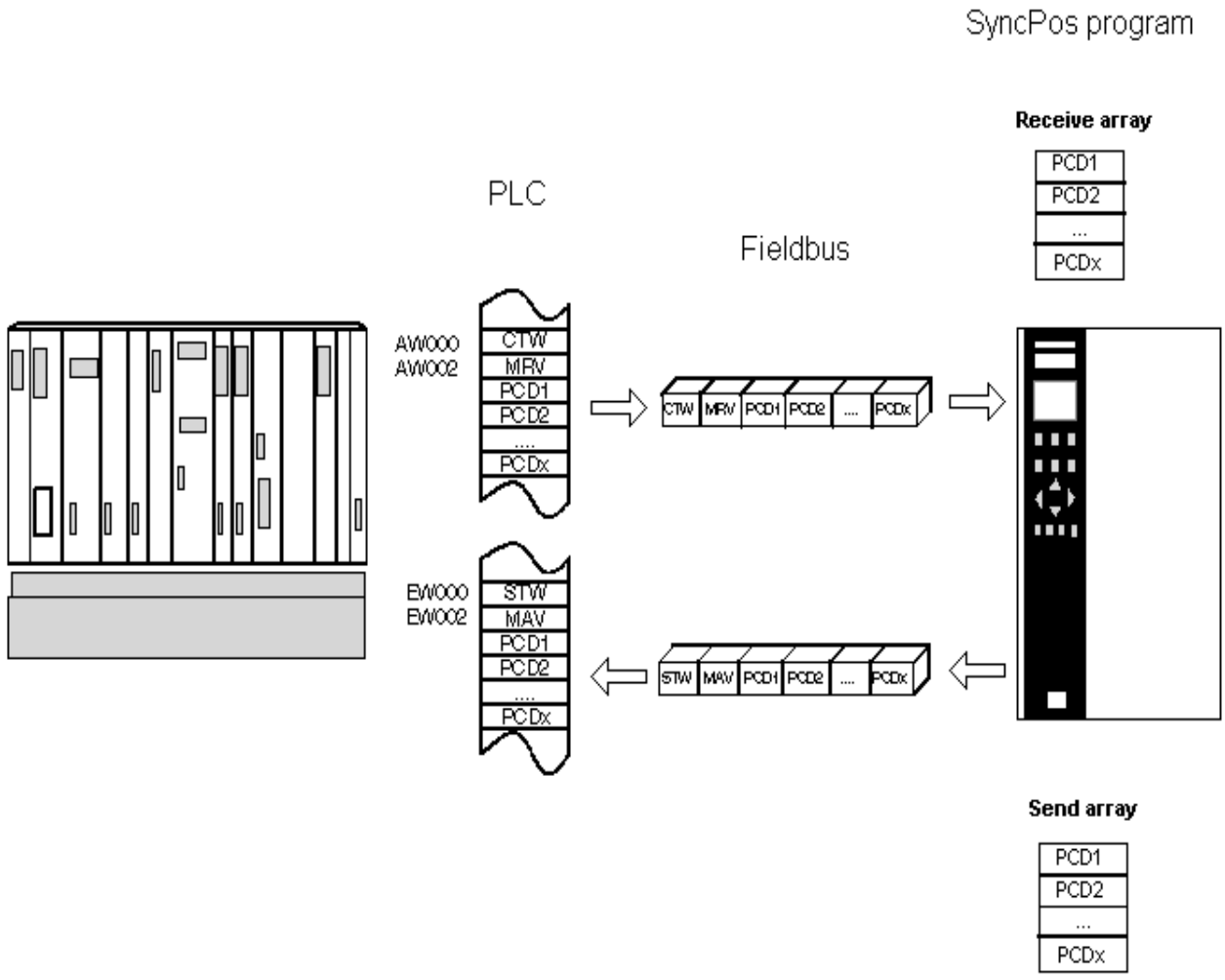
Fieldbus	Number of PCD words
PROFIBUS	PPO type 2 & 4 : 4 PCD's PPO type 5 : 8 PCD's
DeviceNet	Instance 101/151 : 2 PCD's Instance 102/152 : 4 PCD's
Interbus	1 - 7 PCD's

When working with SyncPos the PCD's are not assigned a parameter number by parameter 915 and 916 as is typically done when using a fieldbus option. The SyncPos uses PCD's as free data which can be structured as desired by the programmer.

SyncPos communication commands

The SyncPos option program language includes commands to send and receive PCD's. The command is used to read the contents of the Fieldbus receive register and transfer the data to a SyncPos array and transfer the data to the Fieldbus send register.
The command COMOPTSEND is used to read from a SyncPos array and transfer the data to the Fieldbus send register.

Be aware that when using an Instance/PPO type where the CTW and MRV are included, these words will be ignored. The SyncPos option must maintain control of both the CTW and the MRV in order to regulate position or synchronization. The VLT will still be able to send status word (STW) and main actual value (MAV).



Example 1: Sending and receiving PCD's

```
/ * Program for sending and receiving 8 bytes of
data via a PROFIBUS option using PPO type 2 * /
/ * Definition of arrays * /
DIM send [4], receive [4]
/ * Definition of application parameters * /
LINKGPAR 133 710 "SEND DATA WORD 1" 0 65535 0
LINKGPAR 134 711 "SEND DATA WORD 2" 0 65535 0
LINKGPAR 135 712 "SEND DATA WORD 3" 0 65535 0
LINKGPAR 136 713 "SEND DATA WORD 4" 0 65535 0
/ * Initialize arrays (all elements = 0) * /
i = 1
WHILE (i<=4) DO
     receive [ i ] = 0
     i = i + 1
ENDWHILE
j = 1
WHILE (j<=4) DO
     send [ j ] = 0
     j = j +1
ENDWHILE
/ * Main program loop * /

main:
send [ 1 ] = GET 133
/ * send array, element 1 = value of par. 710 */
send [ 2 ] = GET 134
/ * send array, element 2 = value of par. 711 */
send [ 3 ] = GET 135
/ * send array, element 3 = value of par. 712 */
send [ 4 ] = GET 136
/ * send array, element 4 = value of par. 713 */
COMOPTGET 4 receive
/ * Copy 4 words from comm. Option to receive
array * /
COMOPTSEND 4 send
/ * Copy 4 words from send array to
communication Option * /
/ * Print data of receive array * /
print  "RECEIVED (4 words)", " ", receive [ 1 ],
" ", receive [ 2 ], " ", receive [ 3 ], " ",
receive [ 4 ]
DELAY 2000
GOTO main


/ * End of program * /
```

Types of information exchange

Basically there are two ways of relaying information through the PCD's, byte-wise or bit-wise. Examples of both are given below.

Byte-wise communication

Example 2: Relaying commands to the program

Byte-wise communication is well suited for relaying commands that can be performed one at a time.

"COMMANDWORD" PCD #1	DECIMAL VALUE
STOP	0
JOG+	1
JOG-	2
GO TO POSITION 15000	3

```
DIM Receive [ 4 ]
NOWAIT ON
/ * Let the program cycle through the list of
instructions * /
Main:
  COMOPTGET 4 Receive
  / * copy 4 words from comm. option to
  receive array * /
  IF Receive [ 1 ]  == 0 THEN
     / * Execute a full stop as ordered by the
     PLC * /
     MOTOR  STOP
  ENDIF
  IF Receive [ 1 ] == 1 THEN
     / * Execute a positive motion as ordered by
     the PLC   * /
     CVEL 100
     CSTART
  ENDIF
  IF Receive [ 1 ] == 2 THEN
     / * Execute a negative motion as ordered
     by the PLC * /
     CVEL -100
     CSTART
  ENDIF
  IF Receive [ 1 ] == 3 THEN
    / * Go to position 15000 as ordered by
    the PLC  * /
     POSA 15000
  ENDIF
GOTO Main
/ * End of program * /
```

Example 3: Sending data to the SyncPos option

The program in example 2 is expanded to let the user go to other positions than position 15000 and with other ramps and velocities. This example will illustrate how to receive and use data words.

"TRAJECTORY" PCD's	DECIMAL VALUE
POSITION (PCD #2)	15000
VELOCITY (PCD #3)	50
RAMP (PCD #4)	90

```
DIM Receive [ 4 ]
NOWAIT ON
 / * Let the program cycle through the list of
instructions * /
Main:
  COMOPTGET 4 receive
  / * Copy four words from comm. option to
  receive array * /
  ACC Receive [ 4 ]
  / * Set the acceleration as specified by the
  PLC * /
  DEC Receive
  / * Set the deceleration as specified by the
  PLC * /
  IF Receive [ 1 ] == 0 THEN
     / * Execute a full stop as ordered by the
     PLC * /
     MOTOR  STOP
  ENDIF
  IF Receive [ 1 ] == 1 THEN
     / * Execute positive motion as ordered by
     the PLC * /
     CVEL Receive [ 3 ]
     / * Set the velocity as specified by the
     PLC * /
     CSTART
  ENDIF
  IF Receive [ 1 ] == 2 THEN
     / * Execute a negative motion as ordered
     by the PLC * /
     CVEL -Receive [ 3 ]
     / * Set the velocity as specified by the
     PLC * /
     CSTART
  ENDIF
  IF Receive [ 1 ] == 3 THEN
     VEL Receive [ 3 ]
     / * Set the velocity as specified by the
     PLC * /
     POSA Receive [ 2 ]
     / * Go to the position specified by the
     PLC * /
  ENDIF
GOTO Main
/ * End of program * /
```

■ Bit-wise communication

Bit-wise communication is used if you want to communicate several things through the same PCD at the same time. This is often the case with status information, an example of this is given below.

Example 4: Relaying status information

This example is a further development of example 3. Apart from *receiving* commands and data from the communication option, this time the program will *send* status information to the communication option. The table below shows a list of what status information will be sent and on which bits.

Status word (PCD #1)	**Status bits**											
Status flag	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**...**	**15**	**16**
Motor magnetized	1	X	X	X	X	X	X	X	X	X	X	X
Motor moving	X	1	X	X	X	X	X	X	X	X	X	X
Position reached	X	X	1	X	X	X	X	X	X	X	X	X

```
DIM Receive [ 4 ], Send [ 4 ]
LastCPOS = CPOS   / * Initialize variable * /
NOWAIT ON   / * Let the program cycle through
the list of instructions * /
Main:
  COMOPTGET 4 Receive
  / * Copy 4 words from comm. option to receive
  array * /
  ACC Receive [ 4 ]
  / * Set the acceleration as specified by the
  PLC * /
  IF Receive [ 1 ] == 0 THEN
     / * Execute a full stop as ordered by the
     PLC * /
     MOTOR STOP
  ENDIF
  IF Receive [ 1 ] == 1 THEN
  / * Execute a positive motion as ordered by
  the PLC * /
     CVEL Receive [ 3 ]
     / * Set the velocity as specified by the
     PLC * /
     CSTART
  ENDIF
  IF Receive [ 1 ] == 2 THEN
     / * Execute a negative motion as ordered
     by the PLC * /
     CVEL -Receive [ 3 ]
     / * Set the velocity as specified by
     the PLC * /
     CSTART
  ENDIF
  IF Receive [ 1 ] == 3 THEN
  VEL Receive [ 3 ]
  / * Set the velocity as  specified by the
  PLC * /
  POSA Receive [ 2 ]
  / * Go to the position specified by
  the PLC * /
  ENDIF
  / * Read VLT status word * /
  Temp = GETVLT 534
  / * Calculate status flags * /
  fMotorMagnetized = (Temp & 2048) AND
  (Temp & 512)
  fMotorMoving = (LastCPOS ! = CPOS)
  LastCPOS = CPOS
  fPositionReached = (AXEND & 1)
  / * Calculate status word * /
  SEND [ 1 ] = fMotorMagnetized +
  fMotorMoving*2 + fPositionReached*4
  COMOPTSEND 4 Send
  / * Send status  word * /
GOTO Main
/ * End of program * /
```

Example 5: Creating a "WATCHDOG"

This example is a further development of example 4. To let the external controller (typically a PLC) know that the SyncPos is active, that program execution is not "frozen" at any point and that communication lines are still open between the fieldbus and PLC it is customary to add a WATCHDOG bit in the status word. The WATCHDOG bit is essentially just a bit that continuously toggles on and off.

Note that with this kind of information it is important that the program execution passes through the "WATCHDOG toggle" section on a regular basis. The easiest way of doing that is usually to use the NOWAIT ON instruction in the beginning of the program.

Status word (PDC #1)	Status bits											
Status flag	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**...**	**15**	**16**
Motor magnetized	1	X	X	X	X	X	X	X	X	X	X	X
Motor moving	X	1	X	X	X	X	X	X	X	X	X	X
Position reached	X	X	1	X	X	X	X	X	X	X	X	X
WATCHDOG	X	X	X	1	X	X	X	X	X	X	X	X

```
DIM Receive [ 4 ], Send [ 4 ]
LastCPOS = CPOS
/ * Initialize variable * /
NOWAIT ON
/ * Let the program cycle through the  list of
instructions * /
Main:
  COMOPTGET 4 receive
  / * Copy 4 words from comm. option to receive
  array * /
  ACC Receive [ 4 ]
  / * Set the acceleration as specified by the
  PLC * /
  IF Receive [ 1 ] ==  0 THEN
     / * Execute a full stop as ordered by the
     PLC * /
     MOTOR STOP
  ENDIF
  IF Receive [ 1 ] == 1 THEN
     / * Execute a positive motion as ordered by
     the PLC * /
     CVEL Receive [ 3 ]
     / * Set the velocity as specified by the
     PLC * /
     CSTART
  ENDIF
  IF Receive [ 1 ] == 2 THEN
     / * Execute a negative motion as ordered
     by the PLC * /
     CVEL -Receive [ 3 ]
     / * Set the velocity as specified by the
     PLC * /
     CSTART
  ENDIF
  IF Receive [ 1 ] == 3 THEN
     VEL Receive [ 3 ]
     / * Set the velocity as specified by the
     PLC * /
     POSA Receive [ 2 ]
     / * Go to the position specified by the
     PLC * /
  ENDIF
  / * Read VLT status word * /
  Temp = GETVLT 534
  / * Calculate status flags * /
  fMotorMagnetized = (Temp & 2048) AND
  (Temp & 512)
  fWatchDogBit = NOT fWatchDogBit
  fMotorMoving = (LastCPOS ! =CPOS)
  LastCPOS = CPOS
  fPositionReached =
  (AXEND & 1)
  / * Calculate status word * /
  Send [ 1 ]=fMotorMagnetized +
  fMotorMoving*2 + fPositionReached*4 +
  fWathcDogBit *8
  COMOPTSEND 4 Send
  / * Send status word * /
GOTO Main
/ * End of program * /
```

Conclusion

It should be apparent from the examples provided that using a fieldbus along with the SyncPos option can be a very powerful and flexible means of transferring information between the drive and the system controller. The benefits gained from using a fieldbus and the functional capabilities of the SyncPos option provide a very attractive solution for many drive applications.