

The Application Note is specific to the Unidrive SP

Using the Beckhoff CTNet I/O

Scope

This application note covers basic techniques and concepts required for using the Beckhoff CTNet I/O with the Unidrive SP. Basic SyPT Pro programming techniques and process bus concepts are covered to accomplish remote discrete digital input and output as well as analog input and output over CTNet. Note that these techniques are also applicable to the UD75 or the MD29AN, but this application note illustrates the Beckhoff CTNet I/O coupler used with the Unidrive SP and the SM-Apps module.

This application note assumes that the reader is familiar with basic analog and digital interfacing techniques, which are used to bring each individual analog or digital point to (or from) a Unidrive SP or a Beckhoff I/O terminal.

This application note assumes that the SyPT Pro program development environment is installed on a suitable personal computer (desktop or laptop), and that the reader is familiar with the use of the "Watch Window" feature of this product.

A minimal reference CTNet network was used in the preparation of this application note. This network consisted of a Uni-SP fitted with a SM-Apps module and a Beckhoff Coupler, joined via a short length of CTNet cable and "fitted" with 82 ohm ¼ watt terminating resistors. The PC hosting the SyPT Pro was connected to the SP via a "CT-COMMS-CABLE", forming a RS-485 CT-RTU link.

A "CT-COMMS-CABLE" is available in either a USB or legacy RS-232 version. The RS-232 version was used in the preparation of this application note.

This application note is based on previous work by James Lynch also of Control Techniques, that dealt with using a Beckhoff I/O with the older CTNet-capable products.

Background

Modern machine control systems often have a requirement for numerous and varied analog and discrete digital inputs (and outputs). These systems are commonly built up from a number of building blocks like the Unidrive SP, which has a useful (but not limitless) input and output capability. These building blocks are often tied together with a process bus, such as CTNet. When the needed inputs and outputs exceed what can be serviced by the Unidrive SP(s) present, one or more Beckhoff CTNet I/O couplers can provide the additional remote input-output capability. These blocks are usually referred to as nodes on a network.

This application note illustrates SyPT Pro programming techniques and process bus concepts to make the digital and analog input(s) brought to a Beckhoff I/O input module available to the "custom logic" hosted in a SM-Apps module "fitted" to a Unidrive SP, as well making the Beckhoff I/O digital and analog output(s) available to that same "custom logic".

Background (continued)

Typically, variations of two techniques are used by an industrial process bus to exchange data between nodes, non-cyclic techniques and cyclic techniques.

Non-cyclic approaches are non-deterministic and are driven exclusively by “reads” and “writes” from the consumer / producer of that data.

Cyclic approaches are deterministic and involve dedicated functionality that is usually pre-configured with a software tool to make available one or more “data points” over a dedicated link between the two nodes. The total number of data “points” that can be exchanged with cyclic data transfers are usually less than non-cyclic transfers, but the data transfer throughput of those configured “data points” is both higher and more predictable. The system designer needs to be cognizant of cyclic and non-cyclic messaging and use the communication bandwidth wisely.

One concept that needs coverage at this point, is how a process bus in general, and the Beckhoff I/O in particular, deals with Boolean data types. These data types can take on one of two values which are described as being either On or Off (alternatively, True or False).

As a practical matter, process busses handle “collections” of Boolean types, not individual “bits”. The Beckhoff Buss Coupler takes care of assembling the individual input bits from the Input Modules into an input collection, and the decomposing the output collections into individual “bits” in the Output Modules.

The act of assembling a number of input bits into a usable collection is commonly referred to as “packing” (also called encoding or multiplexing). The act of decomposing a collection of Booleans into individual bits is commonly called “unpacking” (also called decoding or de-multiplexing). And the collection size that the Beckhoff I/O deals with is 16 individual bits (two bytes) and encodes these 16-bits onto 32 bits (four bytes).

REMOTE NETWORK I/O

The high-quality Beckhoff I/O system is available for systems using the

CTNet communications network.

A CTNet port is standard on SM-Applications Plus modules. Beckhoff

systems for CTNet include an I/O bus

coupler and a large variety of snap-on terminal blocks

allowing up to 256 digital inputs or outputs and up to

100 analog inputs and outputs per bus coupler. Up to

64 Beckhoff I/O systems can be attached to a CTNet

network. I/O points can be easily read or written. Contact

Control Techniques for details on the wide range of

available Beckhoff Remote I/O options.



Instructions

This application note can be used in a number of ways:

- One can simply download and open the projects and get the applications running on hardware configured to appear the same to the program logic. This has the advantage of permitting interaction with the watch windows that are provided with the projects.
- One can simply read the application note and study the code for instruction and insight.
- One can attempt to code each project after reviewing the program notes included in the application note for a different hardware configuration and get that working. The program files included would then serve as an “instructor’s solution”

Beckhoff Coupler details

BK7200 Coupler

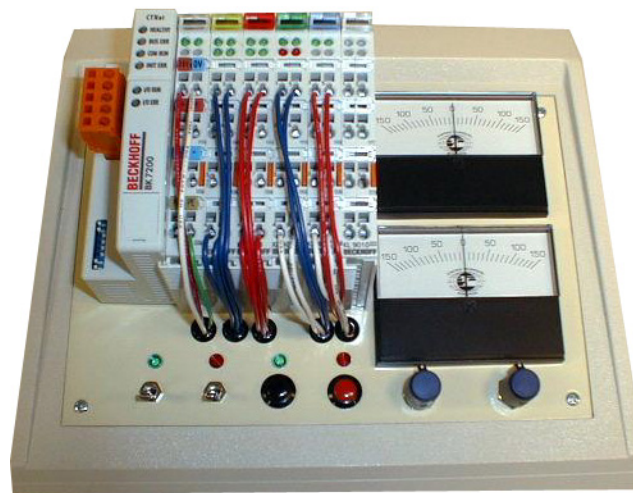
KL1114 Digital INPUT

KL2114 Digital OUTPUT

KL3062 Analog INPUT

KL4032 Analog OUTPUT

KL9010 End Terminal



DEMO Inputs and Outputs

Analog Out 1 (Meter)
Analog Out 2(Meter)
Digital Out 1 (LED)
Digital Out 2 (LED)
Digital Out 3 (LED)
Digital Out 4 (LED)
Digital In 1 (switch)
Digital In 2 (switch)
Digital In 3 (push button)
Digital In 4 (push button)
Analog In 1 Adjust (pot.)
Analog In 2 Adjust (pot.)

Beckhoff CTNet I/O Coupler DEMO / LAB unit

The following SyPT projects are designed to introduce basic concepts first, and then build on those concepts. The initial projects are not reflective of a practical starting for a “real” application, the last two projects are. The introductory material introduced in the initial nine (9) projects contained in this application note tracks the presentation “**Beckhoff Remote IO Basics**”. Refer to this presentation for additional clarification and information.

To obtain this tutorial click here → [Beckhoff Remote IO Basics](#)

To obtain the Example Files click here → [Beckhoff Example Files](#)

Getting Started

Basic Unidrive SP/SM-Apps and Beckhoff CTNet Coupler verification

Project: "SP_BKIO_Test_Basic.CFG"

Watch Window: "SP_BKIO_Test_Basic.wch"

The first step after getting everything assembled is to verify basic CTNet operation between the SM-Apps "fitted" to the UnidriveSP and the Beckhoff coupler. This is a bit more challenging with a minimal SP/Beckhoff network, as the Beckhoff registers that confirm network status are not directly "observable", whereas the UnidriveSP registers are. This project is a demo / tutorial illustrating one solution to this recurring problem. This basic test program generates activity and the watch-window is used to observe the key parameters indicative of proper operation.

Program notes

This is a SyPT Pro demo tutorial program.

This project assumes the presence of a SP at CT-RTU node 1 with a SM-Apps at CTNet0 node 1 fitted in slot 3 and a Beckhoff CTNet I/O at CTNet0 node 64.

This program illustrates the minimal logic needed to show and confirm a SP and Beckhoff I/O on CTNet0 functional.

Three DPL control structures are illustrated.

A block structure is used for the individual Task / Sections that make up the program. A Notes, Initial and Background task are present. A pair of braces {} are used in each case to indicate that the enclosed statements are grouped together.

The most direct way to code an endless loop is used in the Background task. Other ways are available to code this for those that despise the "goto" statement.

An if-then-else structure is used to create a changing value for integer variable i%.

An assignment statement is use make the changing value of i% observable at drive parameter #20.01.

The watch window "SP_BKIO_Activity.wch" is used with the project to monitor key values.

#20.01(1.3:CTNET0_NODE_1)	should be changing (1-1000)
#17.25(1.3:CTNET0_NODE_1)	shows slow / fast CTNet Sync message timing
#17.36(1.3:CTNET0_NODE_1)	shows SP/SM-Apps CTNet0 messages / sec. processed
#00.01(1.3.64:CTNET0_NODE_64)	shows Beckhoff I/O CTNet0 major firmware rev
#00.02(1.3.64:CTNET0_NODE_64)	shows Beckhoff I/O CTNet0 messages / sec. processed
#00.03(1.3.64:CTNET0_NODE_64)	shows Beckhoff terminal K-bus cycles / sec. and indicates how often terminals updated

Non-cyclic read discrete digital input verification

Project: "SP_BKIO_ReadDigitalInNoncyclic.CFG"
Watch Window: "SP_BKIO_ReadDigitalInNoncyclic.wch"

The next step after verifying basic operation would include verifying that any discrete digital input module present on the Beckhoff I/O is functional. Reading digital inputs present on the Beckhoff I/O is easier than writing the digital outputs present on the Beckhoff I/O.

This program illustrates obtaining a collection of input bits (as a word) from the Beckhoff I/O via a CNet read and "unpacking" this collection into individual Boolean variables. A practical program would then make use of the state of these bits elsewhere in the logic.

The DEMO / LAB Beckhoff coupler has four discrete inputs provided by a single KL1114 Input module. The project illustrates a simple technique to make the Beckhoff digital inputs available to the program running in the SM-Apps module.

Program notes

This is a SyPT Pro demo tutorial program.

This program is a derivative of a similar program authored by Jim Lynch of CT for the Uni/UD7x product using the SyPT Workbench.

This program illustrates:

The use of a CNet RDNET command to determine the state of four digital inputs on a Beckhoff I/O module.

The program accomplishes this by reading over CNet the value of Beckhoff I/O parameter #1.00. The result is a 32-bit integer with the lower 16-bits reflecting a collection of the input bits.

Use of a RDNET command also requires insight and use of a NETREPLY command.

This program also illustrates the use of DPL bit operations.

RDNET command format

status% = RDNET(node%, menu%, param%, waitcode%)

status% range	-6 to +6
node% range	0 to 255
menu% range	1 to value set by SP/Options
param% range	0 to value set by SP/Options
waitcode% range	16 to 255 for use illustrated. Units are milliseconds. 16 is the minimum value for reliable operation, 255 is a suggested upper limit.

Note that RDNET with a waitcode% set to zero (0) is legal, but operation differs subtly, and is not illustrated here.

Use of a WRNET command with a waitcode% set to zero (0) would require insight into and use of a NETREPLY command.

NETREPLY command format

result% = NETREPLY(0)

This returns the value for a (previous) CNet read operation.

Also note: the value 0 needs to be passed as an argument for historical purposes.

Non-cyclic write discrete digital output verification

Project: "SP_BKIO_WriteDigitalOutNoncyclic.CFG"

Watch Window: "SP_BKIO_WriteDigitalOutNoncyclic.wch"

The next steps after verifying basic operation would include verifying that any discrete digital output module present on the Beckhoff I/O is functional. Writing the digital outputs present on the Beckhoff I/O (a *set operation*) is more challenging than reading digital inputs present on the Beckhoff I/O (a *get operation*). One must understand and deal with the mask bits as well as the output bits to get this to play.

Proper use of the mask bits have been a conceptual "stumbling block" for many SyPT Pro users, and has resulted in more calls to technical support than any other aspect of using the Beckhoff I/O. Were it not for the mask bits, every change to an individual digital output bit would require a read word, update bit, and write word cycle done in the program running in the SM-Apps module. By providing mask bits to indicate which bits are to be updated along with the new value in a single collection, only a single write is required from the program running in the SM-Apps module to the Beckhoff. The read, update, write cycle is done to the Beckhoff I/O (where it belongs) and where it happens with no intervention by the SyPT Pro programmer.

The DEMO / LAB Beckhoff coupler has four discrete outputs provided by a single KL2114 Output module. The project illustrates a simple technique to make the Beckhoff digital outputs available to the program running in the SM-Apps module. Note that this project illustrates setting all four of the LSB MASK bits to allow any of the discrete digital outputs physically present with a single KL2114 to be updated.

Program notes

This is a SyPT Pro demo tutorial program.

This program is a derivative of a similar program authored by Jim Lynch of CT for the Uni/UD7x product using the SyPT Workbench.

This program illustrates:

The use of a CTNet WRNET command to update a digital output on a Beckhoff I/O using a non-cyclic approach.

The program also shows a simple approach to creating a bit that "toggles", the insertion of this changing bit into a integer, and the combination of a this integer with a constant MASK value that is suitable to be sent to the Beckhoff I/O. This update is accomplished via a non-cyclic WRNET to parameter #2.00 of the Beckhoff I/O.

WRNET command format

status% = WRNET(node%, menu%, param%, value%, dpos%, waitcode%)

status%	range	-6 to +6
node%	range	0 to 255
menu%	range	1 to value set by SP/Options
param%	range	0 to value set by SP/Options
value%	range	0 to +/- 2 ³¹ (signed 32-bit integer range)
dpos%	range	0 to 3 sets decimal point for scaled data
waitcode%	range	16 to 255 for use illustrated. Units are milliseconds. 16 is the minimum value for reliable operation, 255 is a suggested upper limit.

Note that WRNET with a waitcode% set to zero (0) is legal, but operation differs subtly, and is not illustrated here.

Use of a WRNET command with a waitcode% set to zero (0) would require insight into and use of a NETREPLY command.

NETREPLY command format

result% = NETREPLY(0) This returns the value for a (previous) CTNet read operation.

Note: the value 0 needs to be passed as an argument for historical purposes.

Non-cyclic read analog input verification

Project: "SP_BKIO_ReadAnalogInNoncyclic.CFG"

Watch Window: "SP_BKIO_ReadAnalogInNoncyclic.wch"

The next steps after verifying basic operation would include verifying that any Analog input module on the Beckhoff I/O is functional. The DEMO / LAB Beckhoff coupler has two Analog inputs provided by a single KL3062 Analog Input module. The project illustrates a simple technique to make the Beckhoff analog inputs available to the program running in the SM-Apps module.

Program notes

This is a SyPT Pro demo tutorial program.

This program is a derivative of a similar program authored by Jim Lynch of CT for the Uni/UD7x product using the SyPT Workbench.

This program illustrates:

The use of a CTNet RDNET command to determine the value of an Analog Input on a Beckhoff I/O module, the first Analog input in this example.

The program accomplishes this by reading over CTNet the value of Beckhoff I/O parameter #3.00. The result is a 32-bit integer with the lower 16 bits reflecting the value of the Analog Input. Additionally, the program takes this raw value and normalizes it to a zero to 100 percent range.

Use of a RDNET command also requires insight and use of a NETREPLY command.

RDNET command format

status% = RDNET(node%, menu%, param%, waitcode%)

status% range -6 to +6

node% range 0 to 255

menu% range 1 to value set by SP/Options param% range 0 to value set by SP/Options

waitcode% range 16 to 255 for use illustrated. Units are milliseconds. 16 is the minimum value for reliable operation, 255 is a suggested upper limit.

Note that RDNET with a waitcode% set to zero (0) is legal, but operation differs subtly, and is not illustrated here.

Use of a WRNET command with a waitcode% set to zero (0) would require insight into and use of a NETREPLY command.

NETREPLY command format

result% = NETREPLY(0) This returns the value for a (previous) CTNet read operation.

Note: the value 0 needs to be passed as an argument for historical purposes.

Non-cyclic write analog output verification

Project: "SP_BKIO_WriteAnalogOutNoncyclic.CFG"

Watch Window: "SP_BKIO_WriteAnalogOutNoncyclic.wch"

The next steps after verifying basic operation would include verifying that any Analog output module on the Beckhoff I/O is functional. The DEMO / LAB Beckhoff coupler has two Analog outputs provided by a single KL4032 Analog Output module. The project illustrates a simple technique to make the Beckhoff analog outputs available to the program running in the SM-Apps module.

Program notes

This is a SyPT Pro demo tutorial program.

This program is a derivative of a similar program authored by Jim Lynch of CT for the Uni/UD7x product using the SyPT Workbench.

This program illustrates:

The use of a CTNet WRNET command to update a Analog output on a Beckhoff I/O using a non-cyclic approach.

The program also shows a simple approach to creating a value that "ramps", that is suitable to be sent to the Beckhoff I/O. This update is accomplished via a non-cyclic WRNET to parameter #4.00 of the Beckhoff I/O.

WRNET command format

status% = WRNET(node%, menu%, param%, value%, dpos%, waitcode%)

status% range	-6 to +6
node%	range 0 to 255
menu%	range 1 to value set by SP/Options
param%	range 0 to value set by SP/Options
value%	range 0 to +/- 2 ³¹ (signed 32-bit integer range)
dpos%	range 0 to 3 sets decimal point for scaled data
waitcode%	range 16 to 255 for use illustrated. Units are milliseconds. 16 is the minimum value for reliable operation, 255 is a suggested upper limit.

Note that WRNET with a waitcode% set to zero (0) is legal, but operation differs subtly, and is not illustrated here. Use of a WRNET command with a waitcode% set to zero (0) would require insight into and use of a NETREPLY command.

NETREPLY command format

result% = NETREPLY(0) This returns the value for a (previous) CTNet read operation.

Note: the value 0 needs to be passed as an argument for historical purposes.

More advanced (and practical) concepts are next

The previous programs were the simplest possible procedural code that illustrated the key concepts for using the Beckhoff CTNet IO. CTNet read and writes are simple and instructive, but do not offer the needed performance (transfer rate in registers per second) needed by practical programs.

The following programs all use cyclic links configured with the cyclic link editor to get a number of words of data transferred between the BKIO and the SP / SM-Application module, instead of using CTNet reads and writes. Additionally, a more formal INPUT-PROCESS-OUTPUT design pattern is made as "obvious" as possible.

Cyclic read discrete digital input verification

Project: "SP_BKIO_Digital_In_Minimal_Cyclic.CFG"

Watch Window: "SP_BKIO_Digital_In_Minimal_Cyclic.wch"

The next steps after verifying minimal remote I/O operation would include considering more advanced techniques for making some or all of the discrete digital inputs present on the Beckhoff I/O available to the program running on the SM-Apps module. The DEMO / LAB Beckhoff coupler has four discrete inputs provided by a single KL1114 Input module. The project illustrates an advanced technique to make the Beckhoff digital inputs available to the program running in the SM-Apps module.

Program notes

This is a SyPT Pro demo tutorial program.

This program is a derivative of a similar program authored by Jim Lynch of CT for the Uni/UD7x product using the SyPT Workbench.

This program is designed and implemented illustrating an INPUT-PROCESS-OUTPUT design pattern. It is actually too trivial to require this treatment, but this will make more sense when this logic is extended.

This program illustrates the use of a single cyclic link from the Beckhoff to the SP/SM-Apps and four (4) discrete Digital Inputs.

The Digital Inputs are transferred from the Beckhoff IO to the SM-Apps via a configured Cyclic Link for use by the program logic. The program actually reads an "image" of these inputs (packed up) at _S10%.

Cyclic write discrete digital output verification

Project: "SP_BKIO_Digital_Out_Minimal_Cyclic.CFG"

Watch Window: "SP_BKIO_Digital_Out_Minimal_Cyclic.wch"

The next steps after verifying minimal remote I/O operation would include considering more advanced techniques for making some or all of the discrete digital outputs present on the Beckhoff I/O available to the program running on the SM-Apps module. The DEMO / LAB Beckhoff coupler has four discrete outputs provided by a single KL2114 Input module. The project illustrates an advanced technique to make the Beckhoff digital outputs available to the program running in the SM-Apps module.

Program notes

This is a SyPT Pro demo tutorial program.

This program is inspired by similar program(s) authored by Jim Lynch of CT for the Uni/UD7x product using the SyPT Workbench.

This program is designed and implemented using both a STATE design pattern and a INPUT-PROCESS-OUTPUT design pattern.

This program operates in a single state (or condition), the IDLE_STATE and the program generates a "walking bit" pattern that is written to the Beckhoff. This is a trivial example of a STATE design pattern, but this will make more sense as the program is extended.

This program illustrates the use of four (4) discrete Digital Outputs.

The output word "image" is moved to _R10%. This output word is then transferred to the Beckhoff IO from the SM-Apps via a Cyclic Link. The Beckhoff "unpacks" the output word and sets the discrete digital outputs.

Cyclic read analog input verification

Project: "SP_BKIO_Analog_In_Minimal_Cyclic.CFG"

Watch Window: "SP_BKIO_Analog_In_Minimal_Cyclic.wch"

The next steps after verifying minimal remote I/O operation would include considering more advanced techniques for making some or all of the Analog inputs present on the Beckhoff I/O available to the program running on the SM-Apps module. The DEMO / LAB Beckhoff coupler has two analog inputs provided by a single KL3062 Analog Input module. The project illustrates an advanced technique to make the Beckhoff analog inputs available to the program running in the SM-Apps module.

Program notes

This is a SyPT Pro demo tutorial program.

This program is inspired by similar programs authored by Jim Lynch of CT for the Uni/UD7x product using the SyPT Workbench.

This program illustrates:

The use of a cyclic link between the BKIO and the SM-Apps to determine the value of two Analog Inputs on a Beckhoff I/O module.

This cyclic link was defined via the SyPT cyclic link editor.

The program accomplishes this by reading over CTNet the value of Beckhoff I/O parameter #3.00 and #3.01. The result is two signed 32-bit integers.

Additionally, the program takes this raw value and normalizes it to a zero to 100 percent range.

Cyclic write analog output verification

Project: "SP_BKIO_Analog_Out_Minimal_Cyclic.CFG"

Watch Window: "SP_BKIO_Analog_Out_Minimal_Cyclic.wch"

The next steps after verifying minimal remote I/O operation would include considering more advanced techniques for making some or all of the Analog outputs present on the Beckhoff I/O available to the program running on the SM-Apps module. The DEMO / LAB Beckhoff coupler has two analog outputs provided by a single KL4032 Analog Output module. The project illustrates an advanced technique to make the Beckhoff analog outputs available to the program running in the SM-Apps module.

Program notes

This is a SyPT Pro demo tutorial program.

This program is inspired by similar programs authored by Jim Lynch of CT for the Uni/UD7x product using the SyPT Workbench.

This program illustrates: The use of a cyclic link between the SM-Apps to the BKIO to write the value of two Analog Outputs on a Beckhoff I/O module.

This cyclic link was defined via the SyPT cyclic link editor.

The program accomplishes this by generating a changing value and writing this changing value over CTNet via a cyclic link to the Beckhoff I/O parameters #4.00 and #4.01. The result is two 32-bit signed integers.

An output mask is provided to allow evaluation of sending less than all 32 bits. This is provided to allow confirmation that all 32 bits are required to transfer a negative quantity to the BKIO.

Putting it all together

Project: "SP_BKIO_Echo_IO_Cyclic.CFG"

Watch Window: "SP_BKIO_Echo_IO_Cyclic.wch"

This project reflects the material covered earlier and illustrates the demo application that is loaded into the SP Application module and the Beckhoff coupler for the Beckhoff Input Output CNet Demo.

This example is a practical application utilizing a Beckhoff CNet I/O coupler, but it is still "simpler" than many "production" SyPT Pro programs. This problem is simple enough to be solvable without the use of a deterministic task, like a CLOCK Task or a POS Task.

This program illustrates a technique to "echo" the digital and analog inputs being received from the SM-Apps from the Beckhoff CNet Coupler back to the Beckhoff CNet I/O, both without and with small modifications (conditional polarity inversion of the Analog Outputs in this example).

Program notes

This is a SyPT Pro demo tutorial program.

This program is inspired by similar program(s) authored by Jim Lynch of CT for the Uni/UD7x product using the SyPT Workbench.

This program expects three cyclic link(s) configured from node 1 to node 64
This program expects three cyclic link(s) configured from node 64 to node 1

This program is designed and implemented using an INPUT-PROCESS-OUTPUT design pattern.

This program illustrates the use of four (4) discrete Digital Inputs and four (4) discrete Digital Outputs, two (2) Analog Inputs and two (2) Analog Outputs.

This program "echoes" the present state of the Digital and Analog Input(s) to the Digital and Analog Outputs, with a single modification. If toggle switch 1 is "ON" the Analog 1 output is inverted. If toggle switch 2 is "ON" the Analog 2 output is inverted.

The Digital Inputs are transferred from the Beckhoff IO to the SM-Apps via a Cyclic Link for use by the program logic. The program actually reads an "image" of these inputs (packed up) at _S10%.

The Analog Inputs are transferred from the Beckhoff IO to the SM-Apps via two Cyclic Link(s) for use by the program logic. The program actually reads an "image" of these inputs (packed up) at _S11% - _S12%.

The discrete digital inputs are solved and a resulting output word "image" is calculated.

The output word "image(s)" are moved to _R10%-_R12%. This output word is then transferred to the Beckhoff IO from the SM-Apps via Cyclic Link. The Beckhoff "unpacks" the output word and sets the analog and discrete digital outputs. These, in turn, light the LEDs and excite the panel meters on the BKIO Demo.

Project: "SP_BKIO_Digital_Analog_IO_Cyclic.CFG"
Watch Window: "SP_BKIO_Digital_Analog_IO_Cyclic.wch"

This project builds on the material covered earlier and illustrates a demo application that resembles a more practical, non-trivial SyPT program utilizing a Beckhoff CTNet I/O. The logic contained in the SM-Apps is designed to "exercise" the subsections of the reference CTNet / CT-RTU system consisting of a SP/SM-Apps combination, a Beckhoff CTNet I/O coupler and a PC hosting the SyPT Pro. This program is used internally at Grand Island to verify that a SM-Apps CTNet port remains functional over a period of time (overnight) and is dependable.

Like many "real world" SyPT Pro programs, this program has both a non-deterministic "Background Task" and a deterministic "Clock Task".

Program notes

This is a SyPT Pro demo tutorial program.

This program is inspired by similar program(s) authored by Jim Lynch of CT for the Uni/UD7x product using the SyPT Workbench.

This program expects a 10 msec clock period set at #15.11/#16.11/#17.11.

This program expects three cyclic link(s) configured from node 1 to node 64

This program expects three cyclic link(s) configured from node 64 to node 1

This program is designed and implemented using both a STATE design pattern and a INPUT-PROCESS-OUTPUT design pattern.

This program operates in two states, dependent on the condition of the discrete Digital Inputs to the Beckhoff IO. The state transition logic is coded in the CLOCK task, and acted upon in the BACKGROUND task.

This program illustrates the use of four (4) discrete Digital Inputs and four (4) discrete Digital Outputs, two (2) Analog Inputs and two (2) Analog Outputs.

If all the Digital Inputs are OFF, the program is in a "IDLE" state and the program generates a "walking bit" pattern and two "ramping" analog values that are written to the Beckhoff digital and Analog Output(s), respectively.

If any of the Digital Inputs are ON, the program switches to an "ECHO" state and the present state of the Digital and Analog Input(s) are continuously "written" to the Beckhoff IO. The program remains in this state for a period of 5.0 seconds prior to checking that all Digital Inputs are OFF, the condition for a transition back to the "IDLE" state. The 5.0 second period is also reset on any change of the state of any of the digital or analog inputs.

The Digital Inputs are transferred from the Beckhoff IO to the SM-Apps via a Cyclic Link for use by the program logic. The program actually reads an "image" of these inputs (packed up) at _S10%.

The discrete digital inputs are solved and a resulting output word "image" is calculated. The actual logic does not "unpack" the input bits to accomplish this calculation. The input word is solved evaluated each background scan, looking for a "change" and the "value of zero".

The output word "image(s)" is moved to _R10%-_R12%. This output word is then transferred to the Beckhoff IO from the SM-Apps via Cyclic Link. The Beckhoff "unpacks" the output word and sets the analog and discrete digital outputs. These, in turn, light the LEDS and excite the panel meters on the BKIO Demo.

Author: Jim Jeffers
(716)-774-1193

e-mail : <mailto:jim.jeffers@emerson.com>

Appendix

Input Module types supported with firmware 1.04.00:

Digital Input modules:

KL1002, KL1012, KL1032, KL1052, KL1052, KL1104, KL1114, KL1124, KL1154, KL1164, KL1184, KL1194, KL1194, KL1194, KL1212, KL1232, KL1302, KL1302, KL1304, KL1312, KL1314, KL1352, KL1362, KL1382, KL1404, KL1408, KL1414, KL1418, KL1434, KL1488, KL1498, KL1702, KL1712, KL1722, KM1002, KM1004, KM1008, KM1012, KM1014, KM1018

Digital Output modules:

KL2012, KL2022, KL2032, KL2114, KL2124, KL2134, KL2184, KL2212, KL2404, KL2408, KL2424, KL2488, KL2602, KL2612, KL2622, KL2631, KL2641, KL2652, KL2702, KL2712, KL2722, KL2732, KM2002, KM2004, KM2008, KM2022

Analog Input module types:

KL3001, KL3002, KL3011, KL3012, KL3021, KL3022, KL3041, KL3042, KL3044, KL3051, KL3052, KL3054, KL3061, KL3062, KL3064, KL3102, KL3112, KL3122, KL3132, KL3152, KL3162, KL3172, KL3182, KL3201, KL3202, KL3204, KL3311, KL3312, KL3314, KL3351, KL3356, KL3404, KL3408, KL3444, KL3448, KL3454, KL3458

Analog Output module types:

KL4001, KL4002, KL4004, KL4011, KL4012, KL4014, KL4021, KL4022, KL4024, KL4031, KL4032, KL4034, KL4112, KL4132, KL4404, KL4408, KL4414, KL4418, KL4434, KL4438, KL4494

Click below for additional information

[Supported Beckhoff I/O Modules](#)