TaskExpert Reference

IND500x

Weighing Terminal

METTLER TOLEDO

Contents

1 Overview

TaskExpert™ is a fully integrated development environment, which can be used to develop applications for METTLER TOLEDO Industrial Terminals that support Function Blocks programming in a graphical environment. It is intended to allow development of applications that are independent of any specific programming language or syntax. The application is designed, or 'coded,' by specifying the process flow using a rich set of functional blocks. Thus TaskExpert 'programs' resemble flowcharts; they can be generated and manipulated – edited, copied, moved, deleted – and eventually compiled on the PC to generate code. The code is then transferred to the terminal, where it can be executed. This document provides information on the properties and defined usage of these function blocks.

Due to metrological certification requirements, in the seal certification mode, customers will not be able to use customized TE software to modify the interface and functions of metrological certification.

Throughout this document, the reader is referred to examples of command usage, provided in zip files. These are available at the METTLER TOLEDO Partner Portal – log in at http://partner.mt.com, then click on the TaskExpert Information Extranet link. Alternatively, please contact your local METTLER TOLEDO sales representative.

2 Variable Expression Usage

This chapter describes the usage of variables in a project. It includes information on variable types and functions used in expressions.

2.1. Variable Scoping

A namespace is where the task names and the TaskName.subroutine names are unique. Every project has a namespace within which these names are unique.

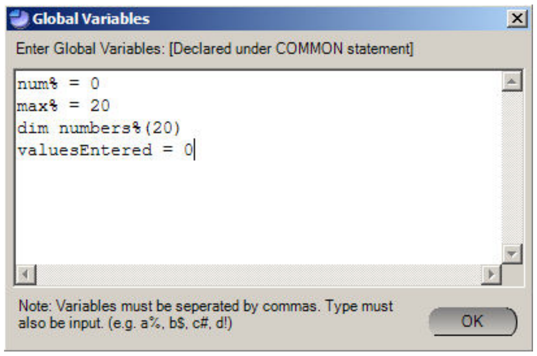
There are four types of variables in TaskExpert projects – Local, Task Global, Shared and Global.

2.1.1. Local variables

A local variable is one that is coded directly inside a block. For example, an expression block with value X = 10 will declare a local variable X of type integer (provided that X is not task Global and Shared Variable). This variable can be used as an integer anywhere in the TaskExpert sequence from that block onwards. Local variables are visible only within the current subroutine. In the final output the local variable names are changed (to provide compact code).

2.1.2. Task Globals

Task Global variables are declared in the Global Variables window (Figure 2-1) by clicking on the Task Globals menu item under View | Variables | Task Globals. A variable declared as a Task Global has scope throughout the declared task in the current project. It cannot be accessed across tasks. In the final output Task Global names are maintained (to provide visibility).



Enter Global Variables: [Declared under COMMON statement]

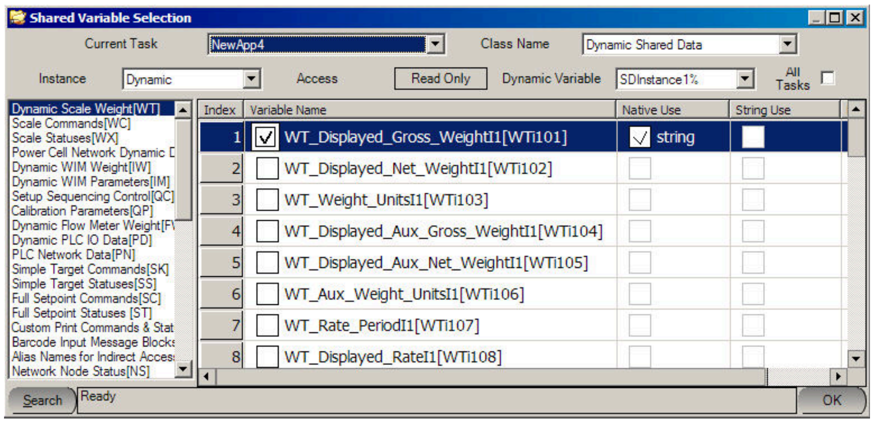
```
num% = 0
max% = 20
dim numbers%(20)
valuesEntered = 0
```

Note: Variables must be seperated by commas. Type must also be input. (e.g. a%, b$, c#, d!)

Figure 2-1: Global Variables

2.1.3. Shared Data variables

A shared variable can be selected from the View | Variables | Shared Variables menu. To use a shared variable it is necessary to select it from the list (i.e. the check box must be in ticked state, as seen in Figure 2-2). Shared variables have a global scope since they may be accessed by any application in the terminal. However, each shared data variable must be selected for use in every task that uses it. Shared variable alias names are not maintained in the final output code.


Shared Variable Selection

Current Task: NewApp4 | Class Name: Dynamic Shared Data
Instance: Dynamic | Access: Read Only | Dynamic Variable: SDInstance1% | All Tasks

Dynamic Scale Weight[WT]
Scale Commands[WC]
Scale Statuses[WX]
Power Cell Network Dynamic D
Dynamic WIM Weight[IW]
Dynamic WIM Parameters[IM]
Setup Sequencing Control[QC]
Calibration Parameters[QP]
Dynamic Flow Meter Weight[FV
Dynamic PLC IO Data[PD]
PLC Network Data[PN]
Simple Target Commands[SK]
Simple Target Statuses[SS]
Full Setpoint Commands[SC]
Full Setpoint Statuses [ST]
Custom Print Commands & Stat
Barcode Input Message Blocks
Alias Names for Indirect Acces
Network Node Status[NS]

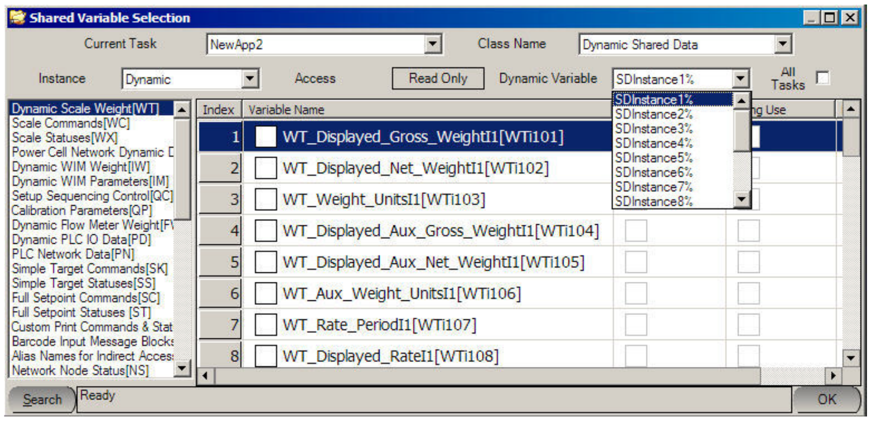
Index	Variable Name	Native Use	String Use
1	✓ WT_Displayed_Gross_WeightI1[WTi101]	✓ string	
2	WT_Displayed_Net_WeightI1[WTi102]		
3	WT_Weight_UnitsI1[WTi103]		
4	WT_Displayed_Aux_Gross_WeightI1[WTi104]		
5	WT_Displayed_Aux_Net_WeightI1[WTi105]		
6	WT_Aux_Weight_UnitsI1[WTi106]		
7	WT_Rate_PeriodI1[WTi107]		
8	WT_Displayed_RateI1[WTi108]		

Search | Ready | OK

Figure 2-2: Shared Variable Selection Window

2.1.4. Dynamic Instances of Shared Variables

Special shared data variables can be used to select the specific shared data variable name at runtime. This shared data is referred to as a dynamic instance. When "Dynamic" is selected as the Instance, an associated instance variable must be selected from the Dynamic Variable list (Figure 2-3). This variable is then used to specify the shared data instance value at runtime. There are fifteen different instance variables that can be used (SDInstance1% - SDInstanceF%).


Shared Variable Selection

Current Task: NewApp2 | Class Name: Dynamic Shared Data
Instance: Dynamic | Access: Read Only | Dynamic Variable: SDInstance1% | All Tasks

SDInstance1%
SDInstance2%
SDInstance3%
SDInstance4%
SDInstance5%
SDInstance6%
SDInstance7%
SDInstance8%

Dynamic Scale Weight[WT]
Scale Commands[WC]
Scale Statuses[WX]
Power Cell Network Dynamic D
Dynamic WIM Weight[IW]
Dynamic WIM Parameters[IM]
Setup Sequencing Control[QC]
Calibration Parameters[QP]
Dynamic Flow Meter Weight[FV
Dynamic PLC IO Data[PD]
PLC Network Data[PN]
Simple Target Commands[SK]
Simple Target Statuses[SS]
Full Setpoint Commands[SC]
Full Setpoint Statuses [ST]
Custom Print Commands & Stat
Barcode Input Message Blocks
Alias Names for Indirect Acces
Network Node Status[NS]

Index	Variable Name		ng Use
1	WT_Displayed_Gross_WeightI1[WTi101]		
2	WT_Displayed_Net_WeightI1[WTi102]		
3	WT_Weight_UnitsI1[WTi103]		
4	WT_Displayed_Aux_Gross_WeightI1[WTi104]		
5	WT_Displayed_Aux_Net_WeightI1[WTi105]		
6	WT_Aux_Weight_UnitsI1[WTi106]		
7	WT_Rate_PeriodI1[WTi107]		
8	WT_Displayed_RateI1[WTi108]		

Search | Ready | OK

Figure 2-3: Shared Variable Selection Window, Dynamic Variable List Showing

Figure 2-4 shows an example of the relationship between the types of variables used in projects.

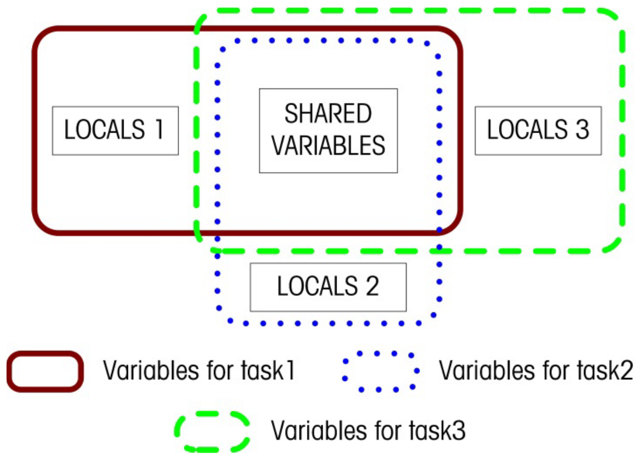


Figure 2-4: Variable Scoping in a Project

2.1.5. Globals

Global variables are declared under COMMON statement. Globals can be declared by clicking View | Variables | Global Variables menu and inserting the variable names inside the Global Variables window. Globals are defined in the startup project and are visible to all tasks in the current project.

2.2. Variable Usage

2.2.1. Variable Types

TaskExpert enables you to represent two fundamental kinds of data: strings and numbers. Number data is further divided into "types." TaskExpert has three numeric data types and one string type.

Integer (A%)	A numeric variable representing a whole number between -2,147,483,648 and +2,147,483,648.
Single precision (A!)	A numeric variable in 32-bit floating point notation between 3.4E-38 and 3.4E+38.
Double precision (A#)	A numeric variable in 64-bit floating point notation between 1.7E-308 and 1.7E+308.
Variable length string (A$)	A list of characters terminated by a 0. Maximum string length is 1000 bytes.

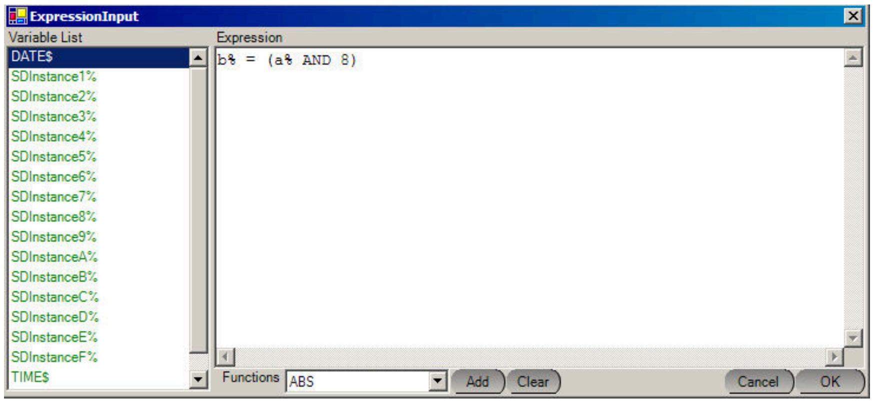
TaskExpert enables you to assign descriptive names to data values, called variables. Variable names can contain up to 16 characters and must begin with a letter. Valid characters are A-Z and 0-9. Variables are case sensitive – for example A$ and a$ are different variables. The last character of the variable name specifies the data type (%, !, #, or $). The maximum number of variables is 1,000 in the IND500x.

Data variables defined in the program are saved in the TaskExpert interpreter until the terminal is powered down.

2.3. Expression Usage

2.3.1. Expression Usage

Expressions can be used in several different Programming Control function blocks. These blocks will have an expression or condition value parameter that is used to input the expression or condition. Selecting this parameter will open an input window where the expression or condition can be entered. Figure 2-5 shows an Expression Input window, with the variable list at left and a drop-down list of functions at the bottom.



```
b% = (a% AND 8)
```

Figure 2-5: Expression Input Window

The Condition Input window (Figure 2-6) has a Variable List on the left and a drop-down list of functions on the right. The functions list contains the advanced mathematical and string commands. The Variable List contains all variables that can be used in the expression, which also includes the dynamic variables (SDInstance1% - SDInstanceF%) and the system variables (TIME$ and DATE$).

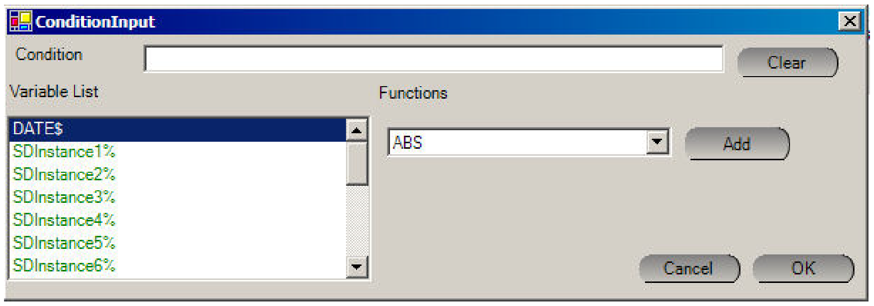


Figure 2-6: Condition Input Window

2.3.2. Precedence of operations

TaskExpert's order of operations has a predefined precedence when evaluating expressions. The following numeric and conditional operators are given in order of precedence.

^	Exponent	=	Equals	>=	Less Than Or Equal
*	Multiply	=	Assign	=>	Greater Than Or Equal
/	Divide	<>	Not Equal	NOT	Not
\	Integer Divide	<	Less Than	AND	And

MOD	Modulus	>	Greater Than	OR	Or
+	Add	<=	Less Than Or Equal	XOR	Exclusive Or
-	Subtract				

AND, OR, and XOR have lower precedence than an assignment operator. Therefore, if you need to assign the results of an AND, OR or XOR operation to a variable, you must put parentheses around the operation.

2.3.3. AND

Used as a logical operator in a decision statement to establish two sets of criteria, both of which must be met. AND can also be used as a bitwise operator between two integer expressions. The AND operator has a lower precedence than assignment operators. Use parentheses around the operation to assign its value to a variable.

Example 1:

```
A>75 AND B<20
```

Example 2:

```
A% = (B% AND 1)
```

2.3.4. OR

Used as a logical operator in a decision statement to establish two possible conditions, either of which must be met. OR can also be used as a bitwise operator between two integer expressions. The OR operator has a lower precedence than assignment operators. Use parentheses around the operation to assign its value to a variable.

2.3.4.1. Example 1:

```
A>75 OR B<20
```

2.3.4.2. Example 2:

```
B% = (A% OR C%)
```

2.3.5. XOR

Used as a logical operator in a decision statement to establish two possible conditions, only one of which can be met. It is used to guarantee that only one variable is true, preventing conflicting options from being true. XOR can be used as a bitwise operator between two integer expressions. The XOR operator has a lower precedence than assignment operators. Use parentheses around the operation to assign its value to a variable.

2.3.5.1. Example 1:

```
A>75 XOR B<20
```

2.3.5.2. Example 2:

```
x%=(4 XOR A%)
```

2.4. Other Math Commands

TaskExpert provides numerous mathematical commands. Using the commands listed in this section, you can perform the following types of mathematical functions on variables or values:

- Trigonometric commands - all angle values are expressed in radians. Multiply the number of radians by (180/π) or approximately 57.3° to convert to degrees.

Command	Usage
ATN	Returns the arctangent of specified numeric expression in radians.
COS	Returns the cosine of a specified angle expressed in radians.
SIN	Returns the sine of a specified angle expressed in radians.
TAN	Returns the tangent of a specified angle expressed in radians.

- Logarithmic and Exponential commands – return the natural logarithm and its complement. Natural logarithms are based on e (approximately 2.718282).

Command	Usage
EXP	Returns e raised to a specified power, where e is the base of natural logarithms.
LOG	Returns the natural logarithm of a numeric expression.

- Conversion & arithmetic operations – convert numbers from one type to another, find absolute value, determine sign, and find the square root. Conversion can be implied by the variable's data type. For example, a#=1 automatically converts the integer 1 to a double precision floating point number.

Command	Usage
ABS	Returns the absolute value of a number.
CINT	Rounds a numeric expression to the closest integer.
CSNG	Converts a numeric expression to a single-precision value.
INT	Returns the largest integer less than or equal to a numeric expression.
SGN	Returns a value indicating the sign of a numeric expression.
SQR	Returns the square root of a numeric expression.

- Random Number commands – generate random numbers.

Command	Usage
RANDOMIZE	Initializes the random-number generator
RND	Returns a single-precision random number between 0 and 1.

2.5. ABS

Returns the absolute value of a number. The absolute value of a number is the magnitude of the number without regard to sign. Absolute values are always positive numbers.

2.5.1. Syntax

```
ABS(numeric-expression)
```

(numeric-expression) = any numeric expression.

2.5.2. Example

```
ABS (45.5-100)
```

Value: 54.5

2.6. ATN

Returns the arctangent of a specified numeric expression in radians. The arctangent is the angle whose tangent is equal to the specified value.

2.6.1. Syntax

```
ATN(numeric-expression)
```

(numeric-expression) = any numeric expression.

2.6.2. Example 1

```
ATN (.75)
```

Value (in radians): 0.6435011

2.6.3. Example 2

```
ATN (.9)
```

Value (in radians): 0.7328151

2.7. CINT

Rounds a numeric expression to the closest integer. The numeric expression can be any number in the range of -32,768 through 32,767.

For positive numbers:

- If the numeric expression contains a fractional part that is less than 0.5, CINT rounds to the next lower integer.
- If the numeric expression contains a fractional part that is greater than or equal to 0.5, CINT rounds to the next higher integer.

For negative numbers:

- If the numeric expression contains a fractional part that is less than 0.5, CINT rounds to the next higher integer.
- If the numeric expression contains a fractional part that is greater than or equal to 0.5, CINT rounds to the next lower integer.

2.7.1. Syntax

```
CINT(numeric-expression)
```

(numeric-expression) = any numeric expression.

2.7.2. Example 1

```
CINT (12.49)
```

Value: 12

2.7.3. Example 2

```
CINT (-12.50)
```

Value: -13

2.8. COS

Returns the cosine of a specified angle expressed in radians.

2.8.1. Syntax

```
COS(angle)
```

Angle = angle expressed in radians

2.8.2. Example

```
pi# = 3.141592654
COS (180 * pi# / 180)
```

Value: -1

2.9. CSNG

Converts a numeric expression to a single-precision value. A single precision numeric variable represents a number of seven or fewer digits plus an exponent. A double precision numeric variable represents a number of eight or more digits plus an exponent. Single-precision and double-precision are also referred to as floating point variables.

2.9.1. Syntax

```
CSNG(numeric-expression)
```

(numeric-expression) = any numeric expression

2.9.2. Example

```
CSNG (975.342151523497)
```

Value: 975.342152

2.10. EXP

Returns e raised to a specified power. The natural logarithm base, e, has a value of approximately 2.71828. The natural logarithm of a number is the power to which the base e must be raised to obtain the number. EXP is the inverse function of the natural log function.

2.10.1. Syntax

```
EXP(numeric-expression)
```

(numeric-expression)= any numeric expression

2.10.2. Example 1

```
EXP (0)
```

Value: 1

2.10.3. Example 2

```
EXP (1)
```

Value: 2.718282

2.11. INT

Returns the integer portion of a specified numeric expression. Rounding does not occur with this command.

For positive numbers:

- The fractional part of the numeric expression is truncated, that is cut-off.

For negative numbers:

- The next lower integer is returned.

2.11.1. Syntax

```
INT(numeric-expression)
```

(numeric-expression) = any numeric expression

2.11.2. Example 1

```
INT (12.54)
```

Value: 12

2.11.3. Example 2

```
INT (-99.4)
```

Value: -100

2.12. LOG

Returns the natural logarithm of a numeric expression. Natural logarithms are based on e, which is approximately 2.718282. The natural logarithm of a number is the power to which the base e must be raised to obtain the number.

2.12.1. Syntax

```
LOG(numeric-expression)
```

(numeric-expression) = any positive numeric expression

2.12.2. Example 1

```
LOG (5)
```

Value : 0.69897

2.12.3. Example 2

```
LOG (EXP(1))
```

Value: 1

2.13. RANDOMIZE

RANDOMIZE specifies a particular initial value or seed value for the random number generator. This seed value is used in specifying the random-number series to be used when the program calls the RND function.

2.13.1. Syntax

```
RANDOMIZE [seed%]
```

seed = [Integer] A number used to initialized the random-number generator.

2.13.2. Example

```
RANDOMIZE
RANDOMIZE a%
```

2.14. RND

RND returns a single-precision random number between 0 and 1. The same sequence of random numbers is generated each time the program runs unless the RANDOMIZE statement was used to specify a different sequence.

RND returns a pseudorandom number which is generated from the seed value using a formula designed to produce numbers that have no pattern or order and appear to be random. Each seed actually creates a fixed sequence of numbers. RANDOMIZE enables you to change the seed value and the sequence generated.

2.14.1. Syntax

```
RND[(n#)]
```

n# = a value that sets how RND generates the next random number.

2.14.2. Example

```
val1% = INT (6*RND + 1)
```

2.15. SGN

Returns a value indicating the sign of a numeric expression. Used to test whether a value is negative, positive, or zero.

- 1 if the expression is positive.
- 0 if the expression is zero.
- -1 if the expression is negative.

2.15.1. Syntax

```
SGN(numeric expression returns)
```

2.15.2. Example

```
SGN (-15)
```

Value: -1

2.16. SIN

Returns the sine of a specified angle expressed in radians.

2.16.1. Syntax

```
SIN(angle)
```

angle = angle expressed in radians

2.16.2. Example

```
pi# = 3.141592654
SIN (90 * pi# / 180)
```

Value: 1

2.17. SQR

Returns the square root of a positive numeric expression.

2.17.1. Syntax

```
SQR(numeric-expression)
```

(numeric-expression) = any numeric expression

2.17.2. Example

```
SQR (25)
```

Value: 5

2.18. TAN

Returns the tangent of a specified angle expressed in radians.

2.18.1. Syntax

```
TAN(angle)
```

angle = angle expressed in radians

2.18.2. Example

```
pi# = 3.141592654
TAN (45 * pi# / 180)
```

Value: 1

2.19. String Operations

TaskExpert's string operations are used to handle string expressions. Each byte in a string expression is treated in one of two ways: 1) as an ASCII character with a value in the range 1 to 127 or 2) as an extended character in the range 128 through 255. The ASCII character set includes uppercase and lowercase letters, numbers, punctuation marks, mathematical symbols, and printer control characters. Strings are terminated by a 0 (null). The maximum usable length of a string is 1000.

The following numeric operators can also be used for string expressions (AND, OR, and XOR can only be used as part of decision statements):

+	Add	AND	And
=	Equals	OR	Or
=	Assign	XOR	Exclusive Or
<>	Not Equal		

TaskExpert's string commands can be used to:

- Convert from a value to a string:

Command	Usage
CHR$	Returns the single-character string corresponding to the specified ASCII code.
HEX$	Returns a string containing the hexadecimal value of a number.
MKI$	Converts integer to string for field.
MKS$	Converts single precision number to string for field.
MKD$	Converts double precision number to string for field.

Command	Usage
OCT$	Returns an octal string representation of a number.
STR$	Returns a string representation of a number.
TIMDAT$	Converts a Julian date number to a string.

- Convert from string to value:

Command	Usage
ASC	Returns the ASCII or extended code value for the first character in a string expression.
CVI	Converts a number string to an integer.
CVS	Converts a number string to a single precision number.
CVD	Converts a number string to a double precision number.
JULDATE	Converts a date-time sting to a double precision julian date number.
VAL	Converts a string representation of a number to a number.

- Parse string data – pad and trim characters:

Command	Usage
LEFT$	Returns a specified number of leftmost characters in a string.
LTRIM$	Removes spaces from the beginning of a string.
MID$	Returns part of a string.
PADC$	Add pad characters to beginning and end of a string.
PADL$	Adds pad characters to the beginning of a string.
PADR$	Adds pad characters to end of a string.
RIGHT$	Returns a specified number of rightmost characters in a string.
RTRIM$	Removes spaces from the end of a string.

- Modify string data – change case or content:

Command	Usage
LCASE$	Converts a string to a lower case.
MSET$	Inserts one string into another string, overwriting the existing characters.
UCASE$	Converts a string to upper case.

- Perform other string functions:

Command	Usage
INSTR	Returns the position of the first occurrence of a string in another string.
LEN	Returns the number of characters in a string or the number of bytes required to store a variable.
SPACE	Returns a string of spaces.

Command	Usage
STRING$	Returns a string of a specified length made up of a repeating character.

2.20. ASC

Returns the ASCII or extended code value of the first character in the specified string expression.

2.20.1. Syntax

```
ASC(stringexpression$)
```

(stringexpression) = [String] Any string expression.

2.20.2. Example

```
ASC("Quiet")
```

Output: 81

The ASCII value of a capital Q is 81.

2.21. CHR$

Returns the single-character string corresponding to the specified ASCII code. Used for characters not easily entered on the keyboard and placed in a string, such as most control characters and graphic characters. The CHR$ commands can generate all 255 characters of the ASCII and extended character sets.

2.21.1. Syntax

```
CHR$(ascii-code%)
```

(ascii-code) = [Integer] ASCII or extended code of the desired character in the range of 1-255.

2.21.2. Example

```
CHR$(65)
```

Output: A

2.22. CKSUM$

This function generates the checksum of a string and returns the checksum in string format. It calculates the checksum by adding the lower 7 bits of each byte in the string and taking the 2's complement. It is used for validating sent and received messages.

2.22.1. Syntax

```
CKSUM$(string1$, [string2$,][string3$,]start%)
```

string1 = [String] Input string with a maximum length of 80 characters.

string2 = [String] Optional input string with a maximum length of 80 characters.

string3 = [String] Optional input string with a maximum length of 80 characters.

Start = [Integer] Character in the string where checksum starts.

2.22.2. Example

```
message$= chr$(2)+"hello world"+chr$(3)
message$= message$+cksum$(message$,1)
```

2.23. COMBITS

The COMBITS command reads the status of the four modem input signals on the COM3 serial port. First open the COM3 serial port using the OPEN command.

2.23.1. Syntax

```
COMBITS(filenumber)
```

(filenumber) = File number used in the OPEN command for the COM3 serial port. COMBITS returns an integer with the following bit values "OR"ed together. The bit value is set to one.

2.23.2. Example

```
a%=combits (1)
```

2.24. CRC$

CRC$ computes a 16 bit CRC on the message text and returns a 4-character string that contains the CRC in ASCII format. The CRC is used primarily with serial communications to ensure that a message is transmitted without errors.

The CRC calculation is a CCITT method that uses an "exclusive OR" hashing method with a lookup table. The CRC calculation starts with the first byte and proceeds sequentially to the last byte of the message text. CRC$ uses the following procedure to calculate and return CRC:

- "Exclusive OR" the high-order byte of the current CRC with the next byte of the message text.
- Use resulting 8-bit value as an index into the lookup table to get 16-bit table value.
- Shift the low-order byte of current CRC to the high-order byte and "exclusive OR" the result with the 16-bit value from step 2. This becomes the new current CRC.
- Go to step 1 and repeat the calculation for each byte of the message.
- "OR" each 4-bit nibble of the 16-bit CRC with a hex 30 to convert the CRC to four printable ASCII characters. Start with low-order byte then convert high order byte last.

The following table is used for the calculating the CRC.

0x0000,	0x1021,	0x2042,	0x3063,	0x4084,	0x50A5,	0x60C6,	0x70E7,
0x8108,	0x9129,	0xA14A,	0xB16B,	0xC18C,	0xD1AD,	0xE1CE,	0xF1EF,
0x1231,	0x0210,	0x3273,	0x2252,	0x52B5,	0x4294,	0x72F7,	0x62D6,
0x9339,	0x8318,	0xB37B,	0xA35A,	0xD3BD,	0xC39C,	0xF3FF,	0xE3DE,

0x2462,	0x3443,	0x0420,	0x1401,	0x64E6,	0x74C7,	0x44A4,	0x5485,
0xA56A,	0xB54B,	0x8528,	0x9509,	0xE5EE,	0xF5CF,	0xC5AC,	0xD58D,
0x3653,	0x2672,	0x1611,	0x0630,	0x76D7,	0x66F6,	0x5695,	0x46B4,
0xB75B,	0xA77A,	0x9719,	0x8738,	0xF7DF,	0xE7FE,	0xD79D,	0xC7BC,
0x48C4,	0x58E5,	0x6886,	0x78A7,	0x0840,	0x1861,	0x2802,	0x3823,
0xC9CC,	0xD9ED,	0xE98E,	0xF9AF,	0x8948,	0x9969,	0xA90A,	0xB92B,
0x5AF5,	0x4AD4,	0x7AB7,	0x6A96,	0x1A71,	0x0A50,	0x3A33,	0x2A12,
0xBDFD,	0xCBDC,	0xFBBF,	0xEB9E,	0x9B79,	0x8B58,	0xBB3B,	0xAB1A,
0x6CA6,	0x7C87,	0x4CE4,	0x5CC5,	0x2C22,	0x3C03,	0x0C60,	0x1C41,
0xEDAE,	0xFD8F,	0xCDEC,	0xDDCD,	0xAD2A,	0xBD0B,	0x8D68,	0x9D49,
0x7E97,	0x6EB6,	0x5ED5,	0x4EF4,	0x3E13,	0x2E32,	0x1E51,	0x0E70,
0xFF9F,	0xEFBE,	0xDFDD,	0xCFFC,	0xBF1B,	0xAF3A,	0x9F59,	0x8F78,
0x9188,	0x81A9,	0xB1CA,	0xA1EB,	0xD10C,	0xC12D,	0xF14E,	0xE16F,
0x1080,	0x00A1,	0x30C2,	0x20E3,	0x5004,	0x4025,	0x7046,	0x6067,
0x83B9,	0x9398,	0xA3FB,	0xB3DA,	0xC33D,	0xD31C,	0xE37F,	0xF35E,
0x02B1,	0x1290,	0x22F3,	0x32D2,	0x4235,	0x5214,	0x6277,	0x7256,
0xB5EA,	0xA5CB,	0x95A8,	0x8589,	0xF56E,	0xE54F,	0xD52C,	0xC50D,
0x34E2,	0x24C3,	0x14A0,	0x0481,	0x7466,	0x6447,	0x5424,	0x4405,
0xA7DB,	0xB7FA,	0x8799,	0x97B8,	0xE75F,	0xF77E,	0xC71D,	0xD73C,
0x26D3,	0x36F2,	0x0691,	0x16B0,	0x6657,	0x7676,	0x4615,	0x5634,
0xD94C,	0xC96D,	0xF90E,	0xE92F,	0x99C8,	0x89E9,	0xB98A,	0xA9AB,
0x 5844	0x4865,	0x7806,	0x6827,	0x18C0,	0x08E1,	0x3882,	0x28A3,
0xCB7D,	0xDB5C,	0xEB3F,	0xFB1E,	0x8BF9,	0x9BD8,	0xABBB,	0xBB9A,
0x4A75,	0x5A54,	0x6A37,	0x7A16,	0x0AF1,	0x1AD0,	0x2AB3,	0x3A92,
0xFD2E,	0xED0F,	0xDD6C,	0xCD4D,	0xBDAA,	0xAD8B,	0x9DE8,	0x8DC9,
0x7C26,	0x6C07,	0x5C64,	0x4C45,	0x3CA2,	0x2C83,	0x1CE0,	0x0CC1,
0xEF1F,	0xFF3E,	0xCF5D,	0xDF7C,	0xAF9B,	0xBFBA,	0x8FD9,	0x9FF8,
0x6E17,	0x7E36,	0x4E55,	0x5E74,	0x2E93,	0x3EB2,	0x0ED1,	0x1EF0

2.24.1. Syntax

```
CRC$(string$)
```

(string) = [String] Input string with a maximum length of 160 characters in the terminal.

2.24.2. Example 1

```
message$= CHR$(2)+"hello world"+CHR$(3)
message$= message$+CRC$(message$)
```

2.24.3. Example 2

```
message$= chr$(2)+"hello world"+chr$(3)
x$=CRC$(message$)
message2$=”happy trails to you”
y$=CRC$(message2$)
z$=CRC$(“a”)
```

2.24.4. Output

```
X$: 9==9
Y$: 060?
Z$: 877<
```

2.25. CVI, CVS, CVD

Convert string variable types, created by either the MKD$, MKI$, or MKS$ commands, to numeric variable types. These commands are used after reading the string representation of a double-precision number in a random-access file that contains records defined by the FIELD statement. Because you cannot store numeric values in random-access files, you must convert numbers to strings before storing them and convert them back to numbers when you read the file.

Command	Returns
CVI	Integer
CVS	Single-precision number
CVD	Double-precision number

2.25.1. Syntax

```
CVI(4-char-numeric-string)
CVS(4-char-numeric-string)
CVD(8-char-numeric-string)
```

(2-byte-numeric string) = 2-byte string variable created by the MKI$ command

(4-byte numeric string) = 4-byte string variable created by the MKS$ command

(8-byte-numeric string) = 8-byte string variable created by the MKD$ command

2.26. DATE$

Sets or returns the terminal system date.

2.26.1. Syntax

DATE$="yyyy-mm-dd"

yyyy-mm-dd = Year, month and day. It is not necessary to enter a leading zero in front of single-digit month or day values.

2.26.2. Example

DATE$ = "2007-10-16"

2.27. EOF

Tests for the end of a file. Returns true (nonzero) if the end of a file has been reached. Used to decide whether to continue processing a file.

2.27.1. Syntax

EOF(filenumber%)

(filenumber) = [Integer] Number of the file to test.

2.27.2. Example

a% = EOF(1)

2.28. EVENTON

Returns the state of the event. A zero value indicates the event is in a "non-triggered" state. A nonzero value is the "triggered" state. You must put quotation marks around the event name.

2.28.1. Syntax

```
EVENTON("event name")
```

"event name" = name of the event

2.28.2. Example

```
EVENTON("SPFEED%")
```

2.29. HEX$

Converts a decimal number (base 10) to a hexadecimal number (base 16).

2.29.1. Syntax

```
HEX$(numeric-expression)
```

(numeric-expression) = any numeric expression.

2.29.2. Example

```
a$ = HEX$(x)
```

2.30. INKEY$

Reads a character from the keyboard or keypad. This commands enables your program to respond to special keys without interrupting program execution. INKEY$ returns a single keystroke from either the keyboard or keypad as a string. As many as 10 keystrokes can be stored in the buffer. If the keystroke was an ASCII character or an extended character, the string is 1-byte.

If there is no keystroke available in the buffer, INKEY$ returns a null string. If you want to retrieve a key and determine if it has one of several values, you must save the keystroke in a variable, as follows:

```
c$=INKEY$
```

2.30.1. Syntax

```
INKEY$
```

2.30.2. Example 1

```
A$=INKEY$
```

2.31. INSTR

Returns the position of the first occurrence of a string in another string. Used for searching text in database fields or for validating user input.

2.31.1. Syntax

```
INSTR(string1$,string2$)
```

string1 = [String] String expression being searched

string2 = [String] String expression that you want to locate

2.31.2. Example

```
DIM prglst$(5)
prglst$(1)="abcdefgh"
prg$="bcd"
x$ = INSTR(prglst$(1),prg$)
```

Output: 2

2.32. JULDATE

The JulDate function returns a double floating-point variable representation of the date and time that it converts from a string representation.

2.32.1. Parameters

The calling parameter is a string in the format "yyyy-mm-dd hh:mm:ss"

- Note: There must be a space between the date and the time.

2.32.1.1. Return Value

The return value is a double floating-point representation of the date and time. The format of the return value is "YYYYDDD.<fractional seconds of the day>". The fractional seconds of the day are calculated as follows:

Total Number of Seconds in a Day:

24h*60m*60s = 86400s

Current Number of Seconds in the Day (using example below):

Current Time = 09:55:23

9h = 32400s

55m = 3300s

23s = 23s

Total Current Number of Seconds = 35723s

Fractional Seconds of the Day = (Current Seconds + 0.5)/Total Seconds

35723.5/86400 = 0.413466435

Note: The 0.5 is added for rounding.

2.32.1.2. Example

```
Datetime$ = "2009-01-29 09:55:23"

MyDatetime# = JulDate(Datetime$)
```

2.32.1.3. Output

```
2009029.413466435
```

2.33. KEYSRC

Reports the source of the latest keystroke that has been read by the application through an INPUT or INKEY$ command.

2.33.1. Syntax

```
KEYSRC
```

2.33.2. Returns

```
0 = No Console Key Entry So Far
1 = Console Key Routing
2 = Console Application Key
3 = Console Softkey
4 = Console Data Entry Line
5 = Console Data Entry Termination
6 = Virtual Console Key
7 = Virtual Console Termination
```

2.34. LCASE$

Converts a string to lower case.

2.34.1. Syntax

```
LCASE$(stringexpression$)
```

stringexpression = [String] Any string expression.

2.34.2. Example

```
A$ = "GOOD MORNING, SUNSHINE"
A$ = lcase$ (a$)
```

Result: A$ = ″good morning, sunshine″

2.35. LEFT$

Returns the specified number of leftmost characters in a string. If you specify a number of characters greater than or equal to the string's length, the entire string is returned.

2.35.1. Syntax

```
LEFT$(stringexpression$,n%)
```

stringexpression = [String] Any string expression.

n = [Integer] Number of characters to return. Range is 0 to 80.

2.35.2. Example

```
a$ = "TaskExpert - IND500x"
b$ = LEFT$(a$, 10)
```

Result: b$ = “TaskExpert”

2.36. LEN

Returns the number of characters in a string or the number of bytes required to store a variable. Used to obtain the length of a string. If a zero is returned, the string is empty.

2.36.1. Syntax

```
LEN(stringexpression$)
```

stringexpression = [String] Any string expression.

2.36.2. Example

```
A$ = "ABC"
A$ = A$ + "C"
LEN(A$)
```

Output: ABCC has length 4

2.37. LOC

Returns the current pointer position within a file that shows where the next read or write operation will take place.

This command can only be used for random access files. LOC returns the next record number after the last record read from or written to the file.

2.37.1. Syntax

```
LOC(filenumber%) #number
```

Filenumber = [Integer] The number of an open file.

#number = The number of records.

Example: LOC(1) = 50

2.38. LOF

Returns the length of a file.

2.38.1. Syntax

```
LOF(filenumber%)
```

Filenumber = [Integer] The number of an open file.

Example: size# = LOF(1)

2.39. LTRIM$

Removes the spaces from the beginning of a string.

2.39.1. Syntax

```
LTRIM$ (stringexpression$)
```

stringexpression = [String] Any string expression.

2.39.2. Example

```
a$ = " 12345"
b$ = LTRIM$(a$)
```

Result: b$="12345"

2.40. MID$

Returns part of a string. The part of the string returned begins at the specified position and contains the given number of characters. If the starting position is greater than the length of the string, a null string is returned. If the number of characters to return is greater than the length of the string, the entire string is returned.

2.40.1. Syntax

```
MID$(stringexpr$,start%[,length%])
```

stringexpr = [String] Any string expression.

start = [Integer] The starting character position to read.

length = [Integer] The number of characters to read.

2.40.2. Example

```
a$ = "Where is Cambridge?"
b$ = MID$(a$, 10, 10)
```

Output: Cambridge?

2.41. MKD$, MKI$, MKS$

Convert numbers to numeric strings that can be stored in FIELD statement string variables. You cannot store numeric values in random-access files. You must convert numbers to strings before storing them. These commands complement the CVI, CVD, and CVS commands which convert the strings back to numbers when you read the file.

Function	Returns
MKI$	2-char string
MKS$	4-char string
MKD$	8-char string

2.41.1. Syntax

```
MKI$(integer-expression%)
MKS$(single-precision-expression!)
MKD$(double-precision-expression#)
```

integer-expression = Any integer number in the range of -2,147,483,648 to +2,147,483,648.

single-precision-expression = Single-precision number in the range of 3.4E-38 to 3.4E+38.

double-precision-expression = Double-precision number in the range of 7E-308 to 7E+308.

2.42. MSET$

Inserts one string into another string at a specified position. Overwrites the existing characters so that the length of the string remains the same.

2.42.1. Syntax

```
MSET$ (string1$, string2$, position%)
```

string1 = [String] String to be changed.

string2 = [String] String to insert.

position = [Integer] Number of character to insert string after.

2.42.2. Example

```
a$="123456789"
b$="abc"
a$=MSET$(a$,b$,3)
```

Output: 123abc789

2.43. OCT$

Converts a number to an octal string.

2.43.1. Syntax

```
OCT$(numeric-expression)
```

numeric expression = Any numeric expression.

2.43.2. Example

```
x=8
b$ = OCT$(x)
```

Output: 8 decimal is 10 octal

2.44. PADC$

Pad the right side and left side of a string, to a specified string length, with a specified string character. The input string is centered in the returned string.

2.44.1. Syntax

```
PADC$(string$, length, padChar$)
```

string = [String] The input string to be padded.

length = Length of the output string.

padchar = [String] Character used as the pad character.

PADC$ returns an input string centered in the output string.

2.44.2. Example

```
a$ = “abc”
b$ = PADC$(a$, 5,”0”)
```

Result: b$ = ”0abc0”

2.45. PADL$

Pad the left side of a string, to a specified string length, with a specified string character.

2.45.1. Syntax

```
PADL$(string$, length, padChar$)
```

string = [String] The input string to be padded.

length = Length of the output string.

padchar = [String] Character used as the pad character.

PADL$ returns an input string right-justified in the output string.

2.45.2. Example

```
a$ = “aBc”
b$ = PADL $(a$, 5,”0”)
```

Result: b$ = ″00aBc″

2.45.3. Example

```
b$ = PADL $(a$, 7,″C″)
```

Result: b$ = ″CCCCaBc″

2.45.4. Example

```
b$ = PADL $(a$, 3,″C″)
```

Result: b$ = ″aBc″

2.46. PADR$

Pad the right side of a string, to a specified string length, with a specified string character.

2.46.1. Syntax

```
PADR$ (string$, length, padchar$)
```

string = [String] The input string to be padded.

length = Length of the output string.

padchar = [String] Character used as the pad character.

PADR$ returns an input string left-justified in the output string.

2.46.2. Example

```
a$ = "aBc″
b$ = PADR$(a$, 5,″0″)
```

Result: b$ = ″aBc00″

2.46.3. Example

```
b$ = PADR$(a$, 7,″C″)
```

Result: b$ = ″aBcCCCC″

2.47. RIGHT$

Returns the specified number of rightmost characters in a string. If you specify a number of characters greater than or equal to the string's length, the entire string is returned.

2.47.1. Syntax

```
RIGHT$(stringexpression$,n%)
```

stringexpression = [String] Any string expression.

n = [Integer] Number of characters to return. The range is 0 to 80.

2.47.2. Example

```
a$ = "TASKEXPERT"
b$ = RIGHT$(a$, 6)
```

Output: EXPERT

2.48. RTRIM$

Removes spaces from the end of the string.

2.48.1. Syntax

```
RTRIM$ (stringexpression$)
```

stringexpression = [String] Any string expression.

2.48.2. Example

```
a$ = "Hello Cambridge ”
b$ = RTRIM$ (a$)
```

Result: b$ = "Hello Cambridge"

2.49. SPACE$

Returns a string of spaces. Used to indent text.

2.49.1. Syntax

```
SPACE$(n%)
```

n = [Integer] The number of spaces to be included in the string. The range is 0 to 80.

2.49.2. Example

```
i% = 5
x$ = SPACE$ (i%)
```

2.50. STR$

Returns a string representation of a number. Used to manipulate a number as a string and to apply string functions to the number for validation and formatting.

2.50.1. Syntax

```
STR$(numeric-expression)
```

numeric expression = Any numeric expression.

2.50.2. Example

```
NUMBER! = 2.5
NUM$ = STR$(NUMBER!)
```

Output: 2.5

2.51. STRING$

Returns a string of a specified length made up of a repeating character. Used to create underlines, rows of asterisks, etc.

2.51.1. Syntax

```
STRING$(length%,{ascii-code% | stringexpression$})
```

length = [Integer] The length of the string.

ascii-code = [Integer] The ASCII code of the repeating character.

stringexpression = [String] The character you want to repeat.

2.51.2. Example

```
STRING$(5, "-")
```

Output: -----

2.52. TIMDAT$

TIMDAT$ converts a double precision floating point Julian Date number to a string: "YYYY-MM-DD HH:MM:SS".

2.52.1. Syntax

```
TIMDAT$(Julian date)
```

2.52.2. Example

```
b# = 974820420
a$ = TIMDAT$(b#)
```

Output: 2000-11-21 10:27:00

2.53. TIME$

Sets or returns the terminal system time.

2.53.1. Syntax

```
TIME$="hh:mm:ss"
```

hh:mm:ss = Hours, minutes and seconds.

2.53.2. Example

```
TIME$ = "10:05:00"
```

2.54. TIMER

Returns a double precision floating point number that contains the elapsed time in seconds since 00:00:00 GMT, January 1, 1970. Used to time the length of specific operations.

2.54.1. Syntax

```
TIMER
```

2.54.2. Example

```
time#=TIMER
```

2.55. UCASE$

Converts a string to upper case.

2.55.1. Syntax

```
UCASE$(stringexpression$)
```

Stringexpression = [String] Any string expression.

2.55.2. Example

```
A$ = "good morning, sunshine”
A$ = ucase$ (a$)
```

Result: A$ = “GOOD MORNING, SUNSHINE”

2.56. VAL

Converts a numeric string to a number. Enables a program to accept numeric input as a string, use various string functions to validate the input, and then convert the input back to a number for use in calculations.

2.56.1. Syntax

```
VAL(stringexpression$)
```

stringexpression = [String] Any numeric string expression.

2.56.2. Example

```
VAL("76")
```

Output: 76

3 Programming Control Commands

3.1. Command Reference

Command	Usage
Call Sub	Branches to a subroutine.
Chain	Enables construction of large application by combining program modules.
ChainCall	Functions like Chain, but remembers location where call was made.
ChainRet	Returns control from the chained program to the location remembered by ChainCall.
Direct Code	Code can be typed directly into this window.
End	Ends a program and closes all files.
Expression	Declares local variables or sets expressions within an application.
For	Repeats the block of statements inside the loop.
GoTo	Branches unconditionally to a specified line.
If	Executes a sub-statement depending on specified conditions.
Notes	Allows addition of comments to the application.
Read Array	Reads a Shared Data byte array (ABv), Bool array (AB1) or Long array (AL) into an array.
ResetIND	Reinitializes the terminal.
Restart	Clears the terminal execution stacks and sends program control to the first line of the current program.
Return	Branches back to the block following the Call Sub block.
Sleep	Suspends program execution for the specified number of milliseconds.
Switch	When a switch case is added, 2 links are shown in the graph. The right link is the default case and the downlink is for the first case.
While	Executes a series of statements as long as a specified condition is true.
WriteArray	Reads an array and copies it into a shared data array.

3.2. Call Sub

Programming Control CallSub

Branches to a subroutine. Used in conjunction with RETURN.

3.2.1. Properties

Function	=	[Function name in the current project] Select a function from the available list.

3.3. Chain

Programming Control Chain

Enables a large application to be programmed, by allowing the application to be split into smaller program modules. CHAIN loads another program and transfers control from the current program to another BASIC program. Variables identified as common variables are accessible by the chained program. CHAIN commands must be placed in the top level of Main, not within a subroutine, IF-THEN, SWITCH, WHILE, or FOR loop.

3.3.1. Properties

File Name	=	The name of the program in the terminal RAMDISK directory to which the current program's controls and variables are to be transferred.

3.4. ChainCall

Programming Control ChainCall

The CHAINCALL command operates the same as a CHAIN command except that it remembers the current program name and line number of the program that is initiating the chaining. After issuing a CHAINCALL, a CHAINRET must be executed before another CHAINCALL can be issued.

3.4.1. Properties

File Name	=	The name of the program in the terminal RAMDISK directory to which the current program's controls and variables are to be transferred.

3.5. ChainRet

Programming Control ChainRet

The CHAINRET command operates the same as a CHAIN command except that it returns control from the chained program to the chaining program at next line after the CHAINCALL.

3.6. Direct Code

Programming Control Direct Code

Code can be typed directly into this window. The code entered is not validated for syntactic correctness and is inserted directly into the output file. For this reason, please be cautious when using the Direct Code block. When local and shared data variables are compiled in the tool, they are assigned a name used by only the output code (i.e. sh000021$). These same compiled names used by the output must be used in the Direct Code block. Note that TaskGlobal variables retain their alias names through the compile process and can also be used in the Direct Code block.

3.7. End

Programming Control End

Ends a program and closes all files. An END statement is executed implicitly at the end of every program.

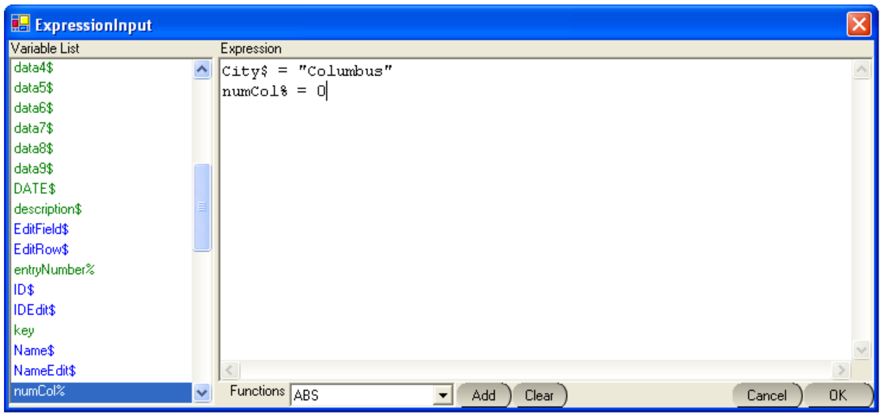
3.8. Expression

Programming Control Expression

The Expression block is used to declare local variables or set expressions within an application. A variable list is included in the ExpressionInput box and updates with every new global task variable and shared data variable that is added to the application. These variables can be accessed by double-clicking on the variable. A Function drop-box is also included to allow easy access to add various expression functions to the expression. Please see the "Expressions" section for definitions of each expression function.

3.8.1. Properties

Value = Any assignment expressions



```
City$ = "Columbus"
numCol% = 0
```

3.9. For

Programming Control For

Repeats the block of statements inside the loop.

3.9.1. Properties

From	=	[Integer] Loop initial value
To	=	[Integer] Loop final value
Step	=	[Integer] Step increment
Counter Variable	=	[Integer, Output/Result variable] Loop counter

3.10. Goto

Programming Control GoTo

Branches unconditionally to a specified line.

3.10.1. Properties

Value	=	Block name in the current subroutine within same level.

3.11. IF

Programming Control IF

Executes the sub-statement depending on specified conditions. The entire IF statement must be contained on one line. The condition is any expression that can be evaluated as true or false. A THEN or ELSE clause can have multiple statements as long as the entire statement is contained on one line.

3.11.1. Properties

Condition	=	Conditional Expression

3.12. Notes

Programming Control Notes

The NOTES block allows the user to add comments throughout an application. The block can be shown or hidden using the "show notes" option available in the Tools > TaskExpert Options > Appearance tab. This block can be placed anywhere within the development frame. NOTES are not executable code and are not included in the compiled code.

3.13. ReadArray

Programming Control ReadArray

Reads a Shared Data byte array (ABv), Bool array (AB1) or Long array (AL) into an array.

3.13.1. Properties

- Note: Indexing is zero- (0) based.

Shared Data Name	=	the name of the shared data that is an array type of byte, bool or long.
Array Name	=	the name of the array where the shared data array data will be stored.
Offset	=	an optional index location into the array to place the shared data array data. Otherwise it will be placed at index 0.
Status	=	Return value. 0 if successful.

3.13.2. Valid combinations

Shared Data Type	DIM Type
Byte Array (ABy)	Integer(%), Float(!), or Double(#)
Boolean Array (ABl)	Integer(%), Float(!), or Double(#)
Long Array (AL)	Double(#)

Invalid combinations will result in a mismatch error.

3.14. ResetIND

Programming Control ResetInd

The ResetIND command re-initializes the terminal by forcing execution through the power-up cycle.

3.15. Restart

Programming Control Restart

Clears the terminal execution stacks and sends program control to the first line of the current program. This command does not affect the variables.

- Note: The Restart command must be issued from within the Main routine in an application. If it is issued from within a subroutine, the command will not be recognized and will create an error

3.16. Return

Programming Control Return

Branches back to the block following the Call Sub block.

3.17. Sleep

Programming Control Sleep

Suspends program execution for the specified number of milliseconds. The terminal timer interrupts every 27.5 milliseconds, so SLEEP can be set up to this accuracy. This command is frequently used to pause a program so the user has time to read the output screen.

3.17.1. Properties

Milliseconds	=	[Integer] The number of milliseconds to suspend program execution.

3.18. Switch

Programming Control Switch

When a switch case is added, 2 links are shown in the graph. The right link is the default case and the downlink is for the first case. The cases entered in the switch case should be separated by comma (,). The code generated would be a set of nested IF-THEN-ELSE statements.

3.18.1. Properties

Cases	=	[String] Set of switch case expressions separated by commas.

3.19. While

Programming Control While

Executes a series of statements as long as a specified condition is true. If the condition is false when the While statement is first encountered, the loop is bypassed and not executed.

3.19.1. Properties

Condition	=	While loop conditional expression

3.20. WriteArray

Programming Control WriteArray

Reads an array and copies it into a shared data array.

3.20.1. Properties

- Note: Indexing is zero- (0) based.

Array Name	=	the name of the array containing data to be copied into shared data.
Shared Data Name	=	the name of the shared data that is an array type of byte, bool or long.
Offset	=	an optional index location into the array to get the shared data array data. Otherwise it will be from index 0.
Status	=	Return value. 0 if successful.

3.20.2. Valid combinations

Shared Data Type	DIM Type
Byte Array (ABy)	Integer(%) with values 0 to 255
Boolean Array (ABl)	Integer(%) wit values 0 or 1

Invalid combinations will result in a mismatch error.

4 Display and Keyboard Commands

4.1. Color Values

The following colors are referenced in this document:

Color Name	Color
Blue	🟦
Cyan	🟦
Green	🟩
White	⬜
Black	⬛
Dark Gray	⬛
Normal Gray	▨
Light Gray	▒
Red	🟥
Orange	🟧
Yellow	🟨

4.2. Command Reference

4.2.1. Display Commands

Command	Usage
Clear Application Display	Clears display objects on the application display window.
Combobox	Displays Combobox on the screen in the application window.

Command	Usage
Control Metrology Line	Turn on or off the metrology line display.
DataGrid	Displays a DataGrid on the screen in the application window.
DispApp	Selects options for the application display.
DispPage	Defines the number of pages to display.
Draw Image	Draws a graphic image in the application window.
Draw Label	Draws a text display object in the application window.
Draw Shared Data	Displays the contents of Shared Data in the application window.
Expopup	Draws an expopup box in the application window.
Focusitem	Determines and returns the object in the display that currently has the focus.
Focuslist	Sets the first-to-last order of the objects for which TaskExpert passes the focus on the application display window.
FocusLisX	Sets the first-to-last order of the objects for which TaskExpert passes the focus on the application display window. Provides more flexibility than the FOCUSLIST function.
Format$	Returns an output string that it derives from a format string and a list of variables or constants.
ImageSD	Draws a variable graphic image in the application window
Popup	Draws a POPUP Box in the application window.
Set Colors	Sets the colors used by the Application window, or resets the window to its default colors.
Set Font	Changes the setting for one of the eight application display fonts.
SmartTrac	Sets the options for the SmartTrac display.
System Message	Writes critical error message to the System error line.
Textbox	Displays the text entry box on the screen in the application window.
Weight Display	Selects options for the weight display.
Yes No Popup	Draws a POPUP Box with two choices in the application window.

4.2.1.1. Softkey Commands

Command	Usage
ExSoftkey	Builds or manipulates an extended Softkey entry in the current Softkey working page.
Softkey Clear	Clears the application Softkey working page.
Softkey Disable	Disables the Softkeys.
Softkey Enable	Enables the Softkeys.
Softkey Home	Copies Softkey home page into the current active Softkey page and displays it on the console display.
Softkey Read	Reads one Softkey page instance and write it to another Softkey page instance.

Command	Usage
Softkey Replace	Replaces the current top page in the Softkey stack with the working page.
Softkey	Builds or manipulates a single Softkey Entry in the current Softkey working page.

4.2.1.2. Keyboard Commands

Command	Usage
Console	Enables the Public Data Entry Line for the operator.
Get Console Enter Key	Routes the Enter key to the current Task Expert console application.
Get Console Scale Keys	Routes the Scale keys to the current Task Expert console application.
Inkey	Allows the application to retrieve one key at a time from the buffered input keys.
KeyEntry	Allows the TaskExpert application to retrieve specific "private" data from the operator console.
Keysrc	Allows the application to retrieve the source of the last input key.
Reset Console Keys	Resets the routing of the Scale keys and Enter Key back to the Control Panel.
Termination Key	Gets the termination key for the last data entry.

4.3. ClearDraw

Display & Keyboard Clear Application Display

Clear display objects on the application display window.

4.3.1. Properties

Index	[Integer, Optional] The index of the display object to clear. If an index value is not specified, then TaskExpert clears all of the objects on the display.
Status	[Integer variable, Output/Result variable, Optional] Return value of this function block.

Return values:

SUCCESS	0

4.4. Combobox

Display & Keyboard Combobox

Displays the Combo Box on the screen in the application window. The application window appears immediately below the Weight/SmartTrac display and above the Softkey display. After the operator moves the focus to the Combo Box on the screen using the navigation keys, he may select one of a number of selections from the Combo Box. The Combo Box is a list of selections that allows the operator to select one from the list. The operator makes his selection via the arrow keys and terminates the selection using the Enter Key. Focus moves to the next entry in the Focus List.

While focus is on the ComboBox, the arrow keys change the current selection. The up and left arrows both move the selection up, and the down and right arrows both move the selection down.

- Note: The TaskExpert variables associated with the Combobox are 40-character strings. The maximum number of characters returned by a Textbox will be 39, since one character is the null terminator.

4.4.1. Properties

Index	[Integer] The index of the display object in the application display. Legal values are 1 to 19. Object 20 is reserved for the Data Entry Line Prompt.
Index Variable Instance	[Integer constant only, Optional] Index of the shared variables to be associated. Legal values are 1-19.
X-Coordinate	[Integer] Position of the text display relative to the top, left-hand corner of the application display window.
Y-Coordinate	[Integer] Position of the text display relative to the top, left-hand corner of the application display window.
Width	[Integer] Width of the combobox item.
Selection List	[Optional] A comma-separated list of selections from which the operator may choose.
Font	[Optional] Font type and size of the display.
Color	[Deafault] Blue. Select color from: Blue, Cyan, Green, White, Black, Dark Gray, Normal Gray, Light Gray, Red, Orange, Yellow
Default	[String, Optional] The selection from Selection List that the COMBOBOX has initially selected when it is first displayed.

Status	[Integer variable, Output/Result variable, Optional]
	Return value of the this function block.
	Return Status:
	`SUCCESS 0`
	`ERR_TOO_MANY_DISPLAY_OBJECTS -1`
	`ERR_OBJECT_NOT_ALLOCATED -4`
	`ERR_DISPLAY_UNICODE_CONVERSION -7`
	ERR_XY_OUT_OF_RANGE -8

VariableName	[Optional] Associated shared variable name, this variable name is mapped to shared variable "tx01XX" where XX=Display Object Index.
AssociatedEvent	[Function name in the current project, Optional] Subroutine to call when data in Combobox is changed. Returned Data: After the operator makes his selection using the navigation keys, TaskExpert returns the number of the selected entry and the selected data to the application, comma-separated, in a Shared Data field, tx0101 through tx0120. The specific Shared Data field corresponds to the object index% specified in the function call. The application can set an event on this Shared Data field so that TaskExpert alerts the application when the data entry is complete. TaskExpert also returns the specific data upon creation of the ComboBox. If the application has registered an event, it will get the event trigger upon creation.

4.5. Console

Display & Keyboard | Console

- Note: In the IND500x, the CONSOLE is set to OFF by default when a TaskExpert application is running

Enable the Public Data Entry Line for the operator, who can then enter data at the operator console using the keypad or keyboard. "Public" means that upon completing the data entry, the operator then selects the function to operate on data entry through a Softkey or another function key. The destination task for the public data entry may be a TaskExpert application, the Control Panel application, or another application task within the terminal. The TaskExpert application retrieves the public data destined for it using the INPUT or INKEY command. The Console command allows the specification of an optional prompt to put on the display for the operator while awaiting the public data entry.

4.5.1. Properties

CONSOLE	Turns off/on the Public Mode Data Entry Line
Default Prompt	[String, Optional] Optional prompt placed on the display for the operator. "" specifies NO prompt for the public data entry line.
Format	[String, Optional] Defines the format of the entered-data. In numeric data-entry-mode, "#nn.dd" is the format specification where nn is max number of numeric digits & dd is decimal point position. In alphanumeric data entry mode, "!ss" is the format specification where ss is maximum number of alphanumeric characters.

4.6. Control Metrology Line

Display & Keyboard | Weight Display

Turns on or off the metrology line display.

4.6.1. Properties

Display	[Optional] Turns on or off the metrology line display. Options: ON, OFF

4.7. DataGrid

Display & Keyboard | DataGrid

Display a DataGrid on the screen in the application window. The DataGrid displays columns of the Standard Tables A0 through A9. It allows the operator to view and edit the database table. The DataGrid takes the entire application window from below status line to the top of the softkeys.

If the application calls the DataGrid so the operator can edit the database table, the application may have an event service routine that enables the application to validate or invalidate the edited data.

4.7.1. Properties

Index	[Integer] Index of the display object in the application display. Legal values are 1 to 19. Object 20 is reserved for Single-Line Data Entry Line Prompt.
Index Variable Instance	[Integer, Optional] Index of the shared variable to be associated. Legal values are 1 to 19.
Variable name	[Optional] Associated shared variable name. This name is mapped to shared variable "tx01XX" where XX = the Display Object Index.
Associated Event	[Function name in the current project, Optional] Subroutine to call when data in DataGrid is changed.
SQL Query	[String] A valid SQL statement.
Examples, IND500x only:	Select 1,tare,=,32,id Same as SQL command: SELECT * FROM A1 WHERE tare = 32 ORDER BY ID Select 2, finefeed,>,2,target Same as SQL command: SELECT * FROM A2 WHERE finefeed > 2 ORDER BY target
HeaderList	[String] Comma-separated list of columns and column headers from the Table that TaskExpert displays in the DataGrid for viewing only. For example, the Header List "ID, Description, Total" defines the 3 columns from the table that will be shown in the DataGrid. The maximum length of the header list is 80 characters.
Column Width	[String] Comma-separated list of column widths that the DataGrid displays on the display. The Column Width list entries correspond to the Header List entries. For example, "0,100,30" essentially hides the ID column but displays the Description column in 100 pixels and the Total column in 30 pixels.
Edit List	[String] Not Used. Defaults to "".
Font	[Optional] Font type and size of the display.
Color	[Optional] Select text color. Options: Black (default)
Y-Coordinate	[Integer] The position of the datagrid relative to the top of the application display window.
Status	[Integer variable, Output/Result variable, Optional]. Return value of this function block. Return Status:
Selected Row	[Optional] When the operator selects a new row, the DataGrid writes the newly selected row's index to tx0154.

Return Status (Status):

```
SUCCESS                               0
SUCCESS – SD PREVIOUSLY USED          1
ERR_TOO_MANY_DISPLAY_OBJECTS         -1
ERR_CANNOT_ACCESS_FILE               -2
ERR_INVALID_HANDLE                   -3
ERR_OBJECT_NOT_ALLOCATED             -4
ERR_INVALID_SHARED_DATA              -5
ERR_CANNOT_SCROLL                    -6
ERR_DISPLAY_UNICODE_CONVERSION       -7
ERR_INVALID_TASK_EXPERT_INSTANCE     -9
```

After the operator completes viewing or editing the Database Table, TaskExpert returns a completion status to the application in the appropriate Shared Data field, tx0101 through tx0120. The specific Shared Data field corresponds to the object index specified in the function call. The application must set an event on this Shared Data field so that TaskExpert alerts the application when the DataGrid operation is complete.

When the operator selects a new row, the DataGrid writes the newly selected row's index to tx0154.

4.8. DispApp

Display Objects DispApp

Select options for the application display. Usually, the application display is always open.

4.8.1. Properties

DISPAPP	OFF, ON. Turns the Application Display off or on.

4.9. DispPage

Display Objects DispApp

Defines the number of pages to be displayed in the IND500x's application display area.

4.9.1. Properties

Page	The number of pages to be displayed. Legal values are 1 to 3. The default is 1.
Status	The return value of the function. Nonzero value indicates a success.

4.10. Draw Image

Display & Keyboard Draw Image

Draw a graphic image in the application window, defining its attributes.

The application window appears immediately below the Weight/SmartTrac display and above the Softkey display. There can be up to 20 display objects in the application display, where an object is a text or graphical display. The (x,y) coordinates for each object are relative to the top, left corner of the application window. The application can overwrite an existing object by executing a display command with the new attributes for the object.

Please remember only one TaskExpert application at a time can interface to the operator console.

4.10.1. Properties

Index	[Integer] Index of the display object in the application display. Legal values are 1 to 19. Object 20 is reserved for Single-Line Data Entry Line Prompt.

X-Coordinate	[Integer] X-Coordinate is the position of the image display relative to the top, left-hand corner of the application display window.
Y-Coordinate	[Integer] Y-Coordinate is the position of the image display relative to the top, left-hand corner of the application display window.
Local Image	[Optional] Image file name to display in TaskExpert development frame. File must be available on local PC.
File Name	[String] Name of the file in the IND500x's memory where the bitmap of the graphic display image resides.
Page	The index of the multipage in the application display. Legal values are 1 to 3. The default is 1.
Status	[Integer variable, Output/Result variable, Optional] Return value of the this function block.

Return Status:

```
SUCCESS                           0
ERR_TOO_MANY_DISPLAY_OBJECTS     -1
ERR_CANNOT_ACCESS_FILE           -2
ERR_INVALID_HANDLE               -3
ERR_OBJECT_NOT_ALLOCATED         -4
ERR_INVALID_SHARED_DATA          -5
ERR_CANNOT_SCROLL                -6
ERR_DISPLAY_UNICODE_CONVERSION   -7
ERR_XY_OUT_OF_RANGE              -8
ERR_INVALID_TASK_EXPERT_INSTANCE -9
```

4.11. Draw Label

Display & Keyboard Draw Label

Draw text display object in the application window, defining its attributes.

The application window appears immediately below the Weight/SmartTrac display and above the Softkey display. There can be up to 20 display objects in the application display, where an object is a text or graphical display. The (x,y) coordinates for each object are relative to the top, left corner of the application window. The application can overwrite an existing object by executing a display command with the new object attributes for the object.

Please remember only one TaskExpert application at a time can interface to the operator console.

4.11.1. Properties

Index	[Integer] Index of the display object in the application display. Legal values are 1 to 19. Object 20 is reserved for Data Entry Line Prompt.
X-Coordinate	[Integer] X-Coordinate is the position of the text display relative to the top, left-hand corner of the application display window.
Y-Coordinate	[Integer] Y-Coordinate is the position of the text display relative to the top, left-hand corner of the application display window.
Text	[Optional] The text of the message
Text ID	[Integer, Optional] Text ID used to fetch text from language database available in the terminal.

Note: Either the Text variable or Text ID must be an input. Both cannot be left empty.

Font	[Optional] Font type and size of the display.
Color	[[Deafault] Blue. Select color from: Blue, Cyan, Green, White, Black, Dark Gray, Normal Gray, Light Gray, Red, Orange, Yellow
Status	[Integer variable, Output/Result variable, Optional] Return value of the this function block. Return Status:

```
SUCCESS                              0
ERR_TOO_MANY_DISPLAY_OBJECTS        -1
ERR_CANNOT_ACCESS_FILE              -2
ERR_INVALID_HANDLE                  -3
ERR_OBJECT_NOT_ALLOCATED            -4
ERR_INVALID_SHARED_DATA             -5
ERR_CANNOT_SCROLL                   -6
ERR_DISPLAY_UNICODE_CONVERSION      -7
ERR_XY_OUT_OF_RANGE                 -8
ERR_INVALID_TASK_EXPERT_INSTANCE    -9
```

4.12. Draw Shared Data

Display the contents of Shared Data in the application window. The application window appears immediately below the Weight/SmartTrac display and above the Softkey display. If the Shared Data field is a "callback" Shared Data field, the function automatically updates display whenever the contents of the display changes.

Please remember only one TaskExpert application at a time can interface to the operator console.

4.12.1. Properties

Index	[Integer] Index of the display object in the application display. Legal values are 1 to 19. Object 20 is reserved for Single-Line Data Entry Line Prompt.
X-Coordinate	[Integer] X-Coordinate is the position of the text display relative to the top, left-hand corner of the application display window.
Y-Coordinate	[Integer] Y-Coordinate is the position of the text display relative to the top, left-hand corner of the application display window.
SD Name	Six-character name of the Shared Data field.
Font	[Optional] Font type and size of the display.
Color	[Deafault] Blue. Select color from: Blue, Cyan, Green, White, Black, Dark Gray, Normal Gray, Light Gray, Red, Orange, Yellow
Format	Defines the format of the displayed data. For numeric data, "#nn.dd" is the format specification where "nn" is max number of numeric digits & "dd" is decimal point position. For alphanumeric string data, "!ss.t" is the format specification where "ss" is maximum number of alphanumeric characters to

	display. "t" defines how to trim blank characters in the string. 1= trim off leading blanks; 2 = trim off trailing blanks; 3 = trim off both leading and trailing blanks. The default is "!0.0" where 0 length implies display all characters in the string and 0 trim implies no trimming of the blank characters.
Status	[Integer variable, Output/Result variable, Optional] Return value of the this function block.

Return Status:

```
SUCCESS                              0
SUCCESS - SD PREVIOUSLY USED         1
ERR_TOO_MANY_DISPLAY_OBJECTS        -1
ERR_CANNOT_ACCESS_FILE              -2
ERR_INVALID_HANDLE                  -3
ERR_OBJECT_NOT_ALLOCATED            -4
ERR_INVALID_SHARED_DATA             -5
ERR_CANNOT_SCROLL                   -6
ERR_DISPLAY_UNICODE_CONVERSION      -7
ERR_XY_OUT_OF_RANGE                 -8
ERR_INVALID_TASK_EXPERT_INSTANCE    -9
```

4.13. ExPopup

Display Objects Expopup AB

Draw an EXPOPUP Box in the application window. The operator can acknowledge the PopUp message by pressing the enter key. The EXPOPUP box does not block TaskExpert application execution, as does the POPUP box function. Therefore, the TaskExpert application can also execute the CLEAR APPLICATION DISPLAY to remove the EXPOPUP box.

The EXPOPUP has a title and two text strings. The box always displays "Press ENTER to continue" at the bottom.

Index	[Integer] Index of the display object in the application display. Legal values are 1 to 19. Object 20 is reserved for Data Entry Line Prompt.
Title	[String] Title of the EXPOPUP box.
Text1	[String] First text message in the EXPOPUP box.
Text2	[String, Optional] Second text message in the EXPOPUP box.
Returned Data	When the operator presses the enter key or the TaskExpert application clears EXPOPUP box with the ClearApplicationDisplay command, TaskExpert writes a trigger to the Shared Data field, tx0101 through tx0120, associated with the display object. The specific Shared Data field corresponds to the object index% specified in the function call. When the application has set a DEFSHR EVENT on this Shared Data field, TaskExpert alerts the application when the EXPOPUP operation is complete by executing the Event Service Routine.

4.14. Set Colors

Display & Keyboard Set Colors

Sets the colors used by the Application window, or resets the window to its default colors.

4.14.1. Properties

Default Fore Color	[Optional] This is the color used for the foreground (text) of display objects. If a color was not specified when they were created. This color will also be used if the color was set to zero when the display object was created.
Default Back Color	[Optional] This is the color used for the back (text) of display objects. If a color was not specified when they were created. This color will also be used if the color was set to zero when the display object was created.
Status	[Integer variable, Output/Result variable, Optional] Return value of this function block.

4.15. Set Font

Display & Keyboard Set Font

Change the setting for one of the eight application display fonts.

4.15.1. Properties

Index	[Integer value only] Available font index. Legal values are 1 to 40.
Font Height	[Integer] New setting for the font height
Status	[Integer variable, Output/Result variable, Optional] Return value of this function block.

4.16. ExSoftkey

Display & Keyboard ExSoftkey

Build or manipulate an extended Softkey entry in the current Softkey working page. The extended entry enables the Softkey Manager to route Softkeys to applications other than TaskExpert. The Softkey working page is a workspace in Shared Data that the TaskExpert application can use before building up a set of Softkeys in a page. Once the application has built all needed Softkeys in the working page, the TaskExpert application can move the working page to the active Softkey page for display.

The ExSoftkey functionality is the same as the Softkey functionality, except that the Softkey hard-codes Softkey destination to be the TaskExpert program. The ExSoftkey function does not hardcode this portion, but sets Softkey destination to the value of SKDest parameter. This feature allows the TaskExpert program build Softkeys that the Softkey Manager routes to the Control Panel

4.16.1. Properties

Soft Key ID	[Integer] Index into the softkeys in the working page. The legal values are 1 to 15.
Soft Key Destination	[Integer] Destination of the softkey signal. The legal values are 1 to 4. 1 refers to the control panel, 4 for TaskExpert programs. Values 2 and 3 are currently reserved.
Mode	Mode of adding the Softkey to the working page. Legal values are OVERWRITE, INSERT and MOVE DOWN, and DELETE.
Key ID	[Integer] Key identifier that the Softkey Manager passes a Softkey to an application. It uniquely identifies the key. If Softkey Destination is set to 1, this value should be less than 100. If Softkey Destination is set to 4, this value should be greater than 100 and less than 65535.
Display Item	[String] Specifies either a text string or a bitmap file that the Softkey Manager uses to draw the key identifier on the display.
Executable File	[String] Specifies the name of an executable file that the Softkey Manager starts when the operator presses this Softkey. Otherwise, the SKM routes the Softkey keystroke to the TaskExpert message window.
Status	[Integer variable, Output/Result variable, Optional] Return value of this function block. Return Value: SUCCESS, CP has taken control of Softkeys Failure, the application terminates with an error.

4.17. Focusitem

Display & Keyboard Focusitem

Determine and return the object in the display that currently has the focus.

4.17.1. Properties

Status	[Integer variable, Output/Result variable, Optional] Return value of the function block. Legal values are 1 to 19. Object 20 is reserved. 0 (zero) signals an error or that nothing has the focus

4.18. Focuslist

Display & Keyboard Focuslist

Set the first-to-last order of the objects for which TaskExpert passes the focus on the application display window.

The focusable objects are DRAW TEXT, DRAW IMAGE, TEXTBOX, COMBOBOX, and DATAGRID. When a DRAW TEXT has the focus, TaskExpert shows it in reverse video. The focused DRAW IMAGE has rectangle drawn around it.

The Enter key normally moves the focus through the display objects according to the focus list. When a DRAW TEXT or DRAW IMAGE has the focus, the up arrow moves the focus to the previous focusable DRAW TEXT / DRAW IMAGE, and the down arrow should move the focus to the next focusable DRAW TEXT / DRAW IMAGE.

4.18.1. Properties

Index	Index of the display object to which to pass the focus.
Status	[Integer variable, Output/Result variable, Optional] Return value of the function block.

Return Values:

SUCCESS	0
ERR_INVALID_HANDLE	-3
ERR_OBJECT_NOT_ALLOCATED	-4

4.19. FocusLisX

Display & Keyboard FocusLisX

Set the first-to-last order of the objects for which TaskExpert passes the focus on the application display window. This function has somewhat more flexibility than the FOCUSLIST function.

The focusable objects are DRAW TEXT, DRAW IMAGE, TEXTBOX, COMBOBOX, and DATAGRID. When a DRAW TEXT has the focus, TaskExpert shows it in reverse video. The focused DRAW IMAGE has rectangle drawn around it.

The Enter key normally moves the focus through the display objects according to the focus list. When a DRAW TEXT or DRAW IMAGE has the focus, the up arrow moves the focus to the previous focusable DRAW TEXT / DRAW IMAGE, and the down arrow should move the focus to the next focusable DRAW TEXT / DRAW IMAGE.

If the application specifies objects in the focus list that do not exist, FOCUSLISX ignores those objects.

4.19.1. Properties

Number of Objects	The number of objects in the focus list to which TaskExpert will pass the focus. Note that when the number of objects in the Focus list is longer than this parameter, TaskExpert only uses this Number of Objects.
First Object	[Integer] The first object in the list to which to pass the focus. This integer value refers to the position in the Object Index parameter list. For example, if the list is 3,5,7, the First Object parameters would be set to 1 to display Index 3, 2 to display Index 5, and 3 to display Index 7.
Object Index	[Integer] List of indexes of the display object to which to pass the focus.
Status	[Integer variable, Output/Result variable, Optional] Return value of the function block.

Return Values:

SUCCESS	0
ERR_TOO_MANY_DISPLAY_OBJECTS	-1
ERR_OBJECT_NOT_ALLOCATED	-4

4.20. Format$

Display & Keyboard FORMAT$

Format$ returns an output string that it derives from a format string and a list of variables or constants. Format$ inserts the variables and constants into the output string based on the format string. It is similar in functionality to the C++ sprintf function, but the format string has the same structure as format string in the TaskExpert PRINT USING statement.

4.20.1. Properties

Format String	[String] String expression containing characters that format the output string. It contains the following format numeric characters: # Digit position . Decimal point position ^ Prints in exponential format _ Space + Sign Use these characters to format string expressions ! Print corresponding characters of string \ \ Print first *n* characters of string, where *n* is the number of blanks between slashes. Other characters are printed as literal data in the output.
Data Parameters	[Optional] List of one or more TaskExpert variables or constants.
Output String	[String variable, Output/Result variable] Formatted string returned from this function block.

4.21. Get Console Enter Key

Display & Keyboard Get Console Enter Key

Route the Enter key to the current TaskExpert console application. Route the alphanumeric keys from the public data entry to the TaskExpert console application when the operator terminates the data entry with the Enter Key. Please remember that only one TaskExpert application at a time can interface to the operator console.

4.22. Get Console Scale Keys

Display & Keyboard Get Console Scale Keys

Route the Scale keys to the current TaskExpert console application. Route the alphanumeric keys from the public data entry to the TaskExpert console application when the operator terminates the data entry with a Scale Key. Please remember that only one TaskExpert application at a time can interface to the operator console.

4.22.1. Properties

Keys to Route	[Optional] ALL, ALPHA ALL routes all keys to the TaskExpert application. ALPHA routes the all the alphanumeric data entry keys to the TaskExpert application. When there is no parameter, route the Scale Function keys to the TaskExpert application.

4.23. ImageSD

Display & Keyboard ImageSD

Draw a variable, changing graphic image in the application window. Upon executing the IMAGESD command, TaskExpert selects the graphic image from a list of up to 6 bitmaps, based on the value in a Shared Data field. TaskExpert automatically switches the display to a new graphic image whenever the Shared Data value changes without further commands from the application program.

The application window appears in the display immediately below the Weight/SmartTrac display and above the Softkey display. There can be up to 20 display objects in the application display, where an object is a text or graphical display. The (x,y) coordinates for each object are relative to the top, left corner of the application window. The application can overwrite an existing object by executing a display command with the new attributes for the object.

Please remember only one TaskExpert application at a time can interface to the operator console.

4.23.1. Properties

Index	[Integer] The index of the display object in the application display. Legal values are 1 to 19. Object 20 is reserved for Single-Line Data Entry Line Prompt.
X-Coordinate	[Integer] The position of the image display relative to the top left-hand corner of the application display window.
Y-Coordinate	[Integer] The position of the image display relative to the top left-hand corner of the application display window.
SD Name	[Integer] The six-character name of the Shared Data field, which must be of type BI, By, US or UL, and be a "callback" field.
Error Filename	[String] The name of the bitmap file which should be displayed when the Shared Data field's value does not match one of the values for which a bitmap is specified in Filename0 - Filename4
Local Filename0	[Optional] File name of bitmap image to display when the Shared Data value is 0. File must be available in Local machine.
Filename0	[String] File name of bitmap image which should be displayed when the Shared Data field's value is 0.
Local Filename1	[Optional] File name of bitmap image to display when the Shared Data value is 1. File must be available in Local machine.
Filename1	[String] File name of bitmap image which should be displayed when the Shared Data field's value is 1.
Local Filename2	[Optional] File name of bitmap image to display when the Shared Data value is 2. File must be available in Local machine.

Filename2	[String] File name of bitmap image which should be displayed when the Shared Data field's value is 2.
Local Filename3	[Optional] File name of bitmap image to display when the Shared Data value is 3. File must be available in Local machine.
Filename3	[String] File name of bitmap image which should be displayed when the Shared Data field's value is 3.
Local Filename4	[Optional] File name of bitmap image to display when the Shared Data value is 4. File must be available in Local machine.
Filename4	[String] File name of bitmap image which should be displayed when the Shared Data field's value is 4.
Status	[Integer variable, Output/Result variable, Optional] Return value of this function block.

Return Status:

SUCCESS	0
ERR_TOO_MANY_DISPLAY_OBJECTS	-1
ERR_CANNOT_ACCESS_FILE	-2
ERR_INVALID_HANDLE	-3
ERR_OBJECT_NOT_ALLOCATED	-4
ERR_INVALID_SHARED_DATA	-5
ERR_CANNOT_SCROLL	-6
ERR_DISPLAY_UNICODE_CONVERSION	-7
ERR_XY_OUT_OF_RANGE	-8
ERR_INVALID_TASK_EXPERT_INSTANCE	-9

4.24. Inkey

Display & Keyboard Inkey

INKEY allows the application to retrieve one key at a time from the buffered input keys. However, the INKEY function returns the key value as an INTEGER instead of as a STRING. It is important to use this function with the Soft keys; since the Soft Key ID's may be defined to be INTEGER values.

4.24.1. Properties

Soft Key ID	[Integer variable, Output/Result variable] Returns an INTEGER key value for a currently outstanding key. If there is no outstanding key, INKEY returns an INTEGER 0.

4.25. KeyEntry

Display & Keyboard Key Entry

The KeyEntry command allows the TaskExpert application to retrieve specific "private" data from the operator console. Using this command, the TaskExpert application retrieves all data entered at the data entry line on the operator console until the operator presses a termination key. The KeyEntry function returns the entered-data in a basic string variable. The TaskExpert application waits suspended until the operator completes the data entry.

In a multi-tasking environment, only one TaskExpert application should interface to the operator console at a time.

4.25.1. Properties

Prompt	[String] Prompt placed on the display for the operator for this specific data entry. "" specifies NO prompt on the data entry line.
Format	[String] Defines the format of the entered-data. In numeric data-entry-mode, "#nn.dd" is the format specification where nn is max number of numeric digits & dd is decimal point position. In alphanumeric data entry mode, "!ss" is the format specification where ss is maximum number of alphanumeric characters.
Font	[Optional] Font type and size of the display.
Color	[Optional] Select text color. Options: Black (default)
Timeout	[Integer, Optional] Parameter that specifies in seconds the time that the operator has to complete the keyboard entry.
Variable	[String variable, Output/Result variable] Basic string variable containing the data entered at the operator console. If a timeout occurs, keyentry$ returns a NULL string & the TermKey() function returns a value of 65535.

4.26. Keysrc

Display & Keyboard Keysrc

The KEYSRC function allows the application to retrieve the source of the last input key.

4.26.1. Properties

Key [Integer variable, Output/Result variable] Retrieves the source of the last input key.

Return Values:

NO_CONSOLE_KEY_ENTRY_SO_FAR	0
CONSOLE_KEY_ROUTING	1
CONSOLE_APPLICATION_KEY	2
CONSOLE_SOFT_KEY	3
CONSOLE_DATA_ENTRY_LINE	4
CONSOLE_DATA_ENTRY_TERMINATION	5
VIRTUAL_CONSOLE_KEY	6
VIRTUAL_CONSOLE_TERMINATION	7

4.27. Popup

Display & Keyboard Popup

Draw a POPUP Box in the application window. The operator must acknowledge the PopUp message by hitting the enter key before TaskExpert program execution continues. The PopUp has a title and two text strings.

4.27.1. Properties

Title	[String] Title of the POPUP box.
Text1	[String] First text message in the POPUP box.
Text2	[String, Optional] Second text message in the POPUP box. Its default value is "Press ENTER to continue". A newline character ("\n") is used to insert a new line into the text. Text2 supports a maximum of 4 lines. The maximum number of characters in each line is 31. The popup expands automatically according to the number of lines.
Status	[Integer variable, Output/Result variable, Optional] Return value of this function block.

4.28. Reset Console Keys

Reset the routing of the Scale keys and Enter Key back to the Control Panel.

4.29. SmartTrac

Sets the options for the SmartTrac display

4.29.1. Properties

Setpoint	[Integer, Optional] Number of the setpoint that drives the SmartTrac display.
Display	[Optional] Turns on or off the SmartTrac display. Options: ON, OFF
Size	[Optional] Sets the size of the SmartTrac display. Options: SMALL, MEDIUM, LARGE

4.30. Soft Key Clear

Clear the application Softkey working page.

4.30.1. Properties

Status	[Integer variable, Output/Result variable, Optional] Return value of this function block. Return Value: SUCCESS, CP has taken control of Softkeys failure, or the application terminates with an error.

4.31. Soft Key Disable

Disable the Softkeys.

4.31.1. Properties

Status	[Integer variable, Output/Result variable, Optional] Return value of this function block. Return Value: SUCCESS, CP has taken control of Softkeys failure, or the application terminates with an error.

4.32. Soft Key Enable

Enable the Softkeys.

4.32.1. Properties

Status	[Integer variable, Output/Result variable, Optional] Return value of this function block. Return Value: SUCCESS, CP has taken control of Softkeys failure, or the application terminates with an error.

4.33. Soft Key Home

Copy Softkey home page into the current active Softkey page and display it on the console display. This restores the soft keys to their original state as at system startup.

4.33.1. Properties

Status	[Integer variable, Output/Result variable, Optional] Return value of this function block. Return Value: SUCCESS, CP has taken control of Softkeys failure, or the application terminates with an error.

4.34. Soft Key Read

Read one Softkey page instance and write it to another Softkey page instance.

4.34.1. Properties

Source Entry	[Integer, Optional] The number of the source Softkey page instance. Legal values are 1 to 8.
Destination Entry	[Integer, Optional] The number of the destination Softkey page instance. Legal values are 1 to 8.
Status	[Integer variable, Output/Result variable, Optional] Return value of this function block.

Return Values:

SUCCESS	0
INVALID PARAMETERS	-1
CP HAS TAKEN CONTROL OF SOFTKEYS	-12

The default SKRead function without any parameters reads the HomePage Softkey page in the Softkey stack into Softkey working page that is instance 8. No function parameters indicate the default command. This default command enables the application to read the current top page in the Softkey stack in order to modify them according to the application's use.

4.34.2. Source and Destination Instances

When the function call specifies a source instance and a destination instance, the SKRead reads the source Softkey page instance and writes it to the destination Softkey page instance.

- Instance 1 is the Home Softkey page.
- Instance 2 is the Current TaskExpert Softkey stack.
- Instance 8 is the application-working page.

4.35. Soft Key Replace

Display & Keyboard Soft Key Replace

Replace the current top page in the Softkey stack with the working page and begin processing the new top.

4.35.1. Properties

Status	[Integer variable, Output/Result variable, Optional] Return value of this function block.

Return Value:

SUCCESS, CP has taken control of Softkeys failure, or application terminates with an error.

4.36. Softkey

Display & Keyboard Softkey

Build or manipulate a single Softkey Entry in the current Softkey working page. The Softkey working page is a workspace in Shared Data that the TaskExpert application can use before building up a set of Softkeys in a page. Once the application has built all needed Softkeys in the working page, the TaskExpert application can move the working page to the active Softkey page for display.

4.36.1. Properties

Soft Key ID	[Integer] Index into the softkeys in the working page. The legal values are 1 to 15. Indexes 1 – 15 refer to Softkeys 1 – 15.
Mode	Mode of adding the Softkey to the working page. Legal values are OVERWRITE, INSERT and MOVE DOWN, and DELETE.
Key ID	[Integer] The key identifier that the SofkKey Manager passes a Softkey to an application. It uniquely identifies the key. It must be larger than 58 and smaller than 65535. If the Softkey is used to finished a console entry, the Softkey value must be smaller than 255.
Display Item	[String] Specifies either a text string or a bitmap file that the Softkey Manager uses to draw the key identifier on the display.
Status	[Integer variable, Output/Result variable, Optional] Return value of this function block. Return Value: SUCCESS, CP has taken control of Softkeys failure, or application terminates with an error.

4.37. System Message

Display & Keyboard System Message

Write critical error message to the System error line. Applications must only use this command for display critical system failures, especially, failures that may cause safety hazards and failures that require immediate operator attention.

4.37.1. Properties

Text	[String] Text of the message to be written to the System error line.
Status	[Integer variable, Output/Result variable, Optional] Return value of this function block.

4.38. Termination Key

Display & Keyboard Termination Key

Get the termination key for the last data entry. The termination key may be an EnterKey, a Scale Key, or a Softkey identifier value. The Softkey identifier may be an INTEGER. After a timeout occurs on KeyEntry$ function, the TermKey() function returns a value of 65535.

4.39. Textbox

Display & Keyboard Text Box

Display the text entry box on the screen in the application window. The application window appears immediately below the Weight/SmartTrac display and above the Softkey display. After the operator moves the focus to the text entry box on the screen using the navigation keys, he may enter data through the keypad or keyboard into the text box. He may terminate the entry with the Enter key.

While the focus is on the textbox, the arrow keys move the cursor within the textbox. The left arrow moves the cursor to the left, and the right arrow both moves the cursor to the right if the format is alphanumeric (i.e. "!ss") or left justified numeric (i.e. "%nn"). If the format dictates right justified numeric entry (i.e. "#nn.dd") the arrow keys have no effect.

When the TEXTBOX receives the focus, it selects the entire contents so the first key pressed replaces the entire contents of the textbox.

Please remember only one TaskExpert application can interface to the operator console.

- Note: The TaskExpert variables associated with the Textbox are 40-character strings. The maximum number of characters returned by a Textbox will be 39, since one character is the null terminator.

4.39.1. Properties

Index	[Integer] Index of the display object in the application display. Legal values are 1 to 19. Object 20 is reserved for Single-Line Data Entry Line Prompt.
Index Variable Instance	[Integer, Optional] Index of the shared variable to be associated. Legal values are 1 to 19.
Variable name	[Optional] Associated shared variable name. This name is mapped to shared variable "tx01XX" where XX = the Display Object Index.
Associated Event	[Function name in the current project, Optional] Subroutine to call when data in Textbox is changed.
X-Coordinate	Position of the text display relative to the top, left-hand corner of the application display window.
Y-Coordinate	Position of the text display relative to the top, left-hand corner of the application display window.
Width	[Integer] Width of the TEXTBOX
Default	Default text displayed in the TEXTBOX
Format	[String, Optional] Defines the format of the entered-data. In numeric data entry mode, "#nn.dd" is the format specification where nn is max number of numeric digits & dd is decimal point position. The numeric data entered into the Text Box appears from right-to-left, filling in behind the decimal point first. In alphanumeric data entry mode, "!ss" is the format specification where ss is maximum number of alphanumeric characters. Data entered into the Text Box appears in left-to-right order. TaskExpert automatically enables alphanumeric data entry through the keypad when the focus comes onto the text box. In alphanumeric password entry mode, "*ss" specifies a format where ss is the maximum number of alphanumeric characters, and entered characters appear as asterisks (*). Data appears in left-to-right order. TaskExpert does not check for ranges and max number of decimal places until the operator presses the Enter key. If there is a problem with the entry the application shows a PopUp error and, after operator acknowledges the PopUp, returns focus to the control.
Font	[Optional] Font type and size of the display.

Color	[Deafault] Blue. Select color from: Blue, Cyan, Green, White, Black, Dark Gray, Normal Gray, Light Gray, Red, Orange, Yellow
Status	[Integer variable, Output/Result variable, Optional] Return value of this function block.

`SUCCESS`	`0`
`ERR_TOO_MANY_DISPLAY_OBJECTS`	`-1`
`ERR_OBJECT_NOT_ALLOCATED`	`-4`
`ERR_DISPLAY_UNICODE_CONVERSION`	`-7`
`ERR_XY_OUT_OF_RANGE`	`-8`

Return Status:

After the operator enters the data and presses either the enter key or one of the navigation keys, TaskExpert returns the data to the application in a Shared Data fields, tx0101 through tx0120. The specific Shared Data field corresponds to the object index% specified in the function call. The application can set an event on the Shared Data field so that TaskExpert alerts the application when the data entry is complete.

TaskExpert also returns the specific data upon creation of the TextBox. If the application has registered an event, it will get the event trigger upon creation.

4.40. Weight Display

Display & Keyboard Weight Display

Select options for the weight display.

4.40.1. Properties

Display	[Optional] Turns on or off the Weight Display. Options: ON, OFF
Size Scale	[Optional] Selects the size of the platform scale weight display. Options: SMALL, MEDIUM, LARGE

4.41. Yes No Popup

Display & Keyboard Popup

Draw a POPUP Box with two choices in the application window. The operator must acknowledge the PopUp message by selecting one of the choices before TaskExpert program execution continues.

4.41.1. Properties

Title	[String] Title of the POPUP box.
Text1	[String] First text message in the POPUP box.
Text2	[String] Second text message in the POPUP box. Its default value is "Press ENTER to continue". A newline character ("\n") is used to insert a new line into the text. Text2 supports a maximum of 4 lines.

Choice 1	[String] The text displayed on the left button. A maximum of 4 characters can be displayed.
Choice 2	[String] The text displayed on the right button. A maximum of 4 characters can be displayed.
Status	[Integer variable, Output/Result variable] Returns a value indicating which choice was selected. 0 = Choice 1 1 = Choice 2

4.42. Key Codes, IND500x

The following table shows the mapping of the keypad keys for the IND500x.

Keypad	Key Code
SK1	1
SK2	2
SK3	3
SK4	4
SK5	5
1	49
2	50
3	51
4	52
5	53
6	54
7	55
8	56
9	57
0	48
.	46
Shift	25
C (Clear)	8
Enter	13
Left Arrow	16
Right Arrow	18
Up Arrow	17
Down Arrow	19
Zero	30
Tare	24
Print	23

5 Scale Commands

- Note: For the Scale property in commands, the only legal value for IND500x is 1. For the Node property in common commands, the only legal value for the IND500x is 0.

5.1. Command Reference

- Notes: References to "specified" or "selected" scale refer only to the IND500x's single scale.

Command	Usage
Clear Tare	Clears tare for selected scale.
ConvertWt	Convert weight from one weight-unit to another weight-unit.
Define Rate	Define the rate calculation parameters for the specified scale.
Filter	Set filter parameters for scale and restart filtering.
Get Average Weight	Calculate the average net weight on a scale over the specified amount of time in milliseconds.
Get Rate	Get the current rate for the specified scale as a double value.
Get Rate$	Get the current rate for the specified scale as a string value.
Get Setpoint	Get the feed status of a setpoint.
Get Weight	Get the legal-for-trade weight and status for the selected scale and node.
Setpoint Setup	Setup the control values for a setpoint.
Setpoint Target	Set a new coincidence target for this setpoint.
Set Units	Select the primary or secondary units operation for a scale.
Stop Setpoint	Stop the current feed for this setpoint.
Switch Resolution	Switch the resolution of the scale weight between its standard resolution and its "x10" resolution.
Tare	Tare selected scale.
WTSTR$	Convert floating-point weight from a floating-point representation to a string representation.
Zero Scale	Attempt to Zero the specified scale.

5.2. Clear Tare

Scale Clear Tare

5.2.1. Properties

Scale	[Integer] Selects the scale. Legal values are: 1.
Node	[Integer, Optional] Selects the node in the cluster. Legal values are: 0. 0 or non-existent parameter refers to the local node.
Tare Status	[Integer variable, Output/Result variable, Optional] Return value of this function block.

Return Status:

TARE_STATUS	
TARE_COMPLETED_SUCCESSFULLY	0
TARE_IN_PROGRESS	1
SCALE_IN_MOTION_DURING_TARE	2
PUSHBUTTON_TARE_NOT_ENABLED	3
PROGRAMMABLE_TARE_NOT_ENABLED	4
CHAIN_TARE_NOT_PERMITTED	5
ONLY_INCREMENTAL_CHAIN_TARE_PERMITTED	6
TARE_NOT_IN_ROUNDED_INCREMENT_VALUE	7
TARE_VALUE_TOO_SMALL	8
TARING_WHEN_POWER_UP_ZERO_NOT_CAPTURED	9
TARING_OVER_CAPACITY	10
TARING_UNDER_ZERO	11
TARE_VALUE_EXCEEDS_LIMIT	12
MUST_CLEAR_TARE_AT_GROSS_ZERO	13
INVALID_TARE_FUNCTION_PARAMETER	98
CANNOT_ACCESS_TARE_SD_TRIGGER	99

5.3. ConvertWt

Scale ConvertWT

Convert weight from one weight-units to another weight-units. Calculate the rounded weight according to the scale increment-size settings for a particular scale.

5.3.1. Properties

OldWt	[Double] The weight value to be converted.
OldUnits	The old weight units. Options: None, pounds, kilograms, grams, metric Tons, tons, oz
NewUnits	The new weight units. Options: None, pounds, kilograms, grams, metric Tons, tons, oz
Scale	[Integer, Optional] Selects the scale. Legal values are: 1. If used, the output weight value will be converted into an increment size consistent with the selected scale. When there is no scale% parameter, ConvertWt rounds the new weight to an increment size of .001.
NewWt	[Double variable, Output/Result variable] Return value of this function block.

5.4. Define Rate

Scale Define Rate

Define the rate calculation parameters for the specified scale.

5.4.1. Properties

Scale	[Integer] Selects the scale. Legal values are: 1.
Node	[Integer, Optional] Selects the node in the cluster. Legal values are: 0. 0 or non-existent parameter refers to the local node.
Measurement	Specifies how often the weight is calculated. Options: one second, five seconds, half-second
Sample Window	[Integer] Specifies the number of intervals over which the rate is calculated. Legal values are 1 to 60 intervals.
Status	[Integer variable, Output/Result variable, Optional] Status of this function block. Please refer to the status definition under the Set Units function block.

5.5. Filter

Scale Filter

Set filter parameters for scale. Restart filtering.

5.5.1. Properties

Scale	[Integer] Selects the scale. Legal values are: 1.
Node	[Integer, Optional] Selects the node in the cluster. Legal values are: 0. 0 or non-existent parameter refers to the local node.
Poles	[Optional] Selects the number of poles for the low pass filtering. Legal values are: 2, 4, 6, 8. The number of filter poles defines the band slope; the transition slope describes the rate of change of the attenuation once outside the pass band. The steeper the slope, the more effective a filter is at rejecting a disturbance that is near the corner frequency. The price is delay; the steeper the slope, the longer the settling time.
Low Pass Frequency	[Double] Selects the lowpass filtering corner frequency. Legal values are 0.1 to 9.9. The pass band extends from 0 Hz to the corner frequency. The low pass filter accepts the frequencies within this low-pass range with little or no attenuation, but attenuates frequencies above the pass band according to the slope of the transition band. The scale is measuring the static weight signal, so it is tempting to make the corner frequency very low to reject all "noise". However, the narrower the pass band, the longer the delay or settling time before we get the final value. As the corner frequency is increased, the scale will settle faster, but will also allow more noise through.

Notch Filter Frequency	[Double, Optional] Selects the notch filter frequency. Legal values are 0 to 99. It is only applicable to Analog Load Cell scales. The notch filter provides attenuation at a single frequency, and little or no attenuation at other frequencies. It is useful in special cases where there is a single noise frequency, such as an A/C power line frequency, near or below the corner frequency of the low pass filter. Note: Do <u>NOT</u> put a value of 0.0 in for the notch frequency.
Status	[Integer variable, Output/Result variable, Optional] Return status of this function block.

Return Status:

BASIC_FUNCTION_STATUS	
FUNCTION_COMPLETED_SUCCESSFULLY	0
FUNCTION_IN_PROGRESS	1
FUNCTION_NOT_ENABLED	2
FUNCTION_NODE_NOT_ONLINE	3
INVALID_BASIC_FUNCTION_PARAMETER	98
CANNOT_ACCESS_FUNCTION_SD_TRIGGER	99

5.6. Get Average Weight

Calculate the average net weight on a scale over the specified amount of time in milliseconds. GETAVGWT samples the weight every 100 milliseconds. For example, if the specified time is 2000 milliseconds, GETAVGWT will take 21 evenly spaced samples of the weight to calculate the average weight.

5.6.1. Properties

Scale	[Integer] Selects the scale. Legal values are: 1.
Node	[Integer] Selects the node in the cluster. Legal values are 0. 0 refers to the local node.
Duration	[Integer] The length of time over which the TaskExpert calculates the average weight.
Average Net Weight	[String or double variable, Output/Result variable] The average net weight of the selected scale. The data type depends on the data type of the variable. If the data type is STRING, GETAUXWT returns a string representation. If the data type is DOUBLE, GETAUXWT returns a floating point representation of the net weight.

5.7. Get Rate

Get the current rate for the specified scale as a double value.

5.7.1. Properties

Scale	[Integer] Selects the scale. Legal values are 1.
Node	[Integer, Optional] Selects the node in the cluster. Legal values are 0. 0 or non-existent parameter refers to the local node.
Rate	[String, Output/Result variable] Returns the rate in DOUBLE floating point format.

5.8. Get Rate$

Scale	GetRate$

Get the current rate for the specified scale as a string value.

5.8.1. Properties

Scale	[Integer] Selects the scale. Legal values are 1.
Node	[Integer, Optional] Selects the node in the cluster. Legal values are 0. 0 or non-existent parameter refers to the local node.
Rate	[Double variable, Output/Result variable] Returns the rate in STRING format.

5.9. Get Setpoint

Scale	Get Setpoint

Get the feed status of a setpoint.

5.9.1. Properties

- In the IND500x, the Tolerance bits are only updated when the Target Mode is set to Over/Under.

Setpoint	[Integer] The number of the setpoint that this function initializes. Legal values are: 1.
Node	[Integer, Optional] Selects the node in the cluster. Legal values are 0. 0 or non-existent parameter refers to the local node.
Setpoint Status	[Integer variable, Output/Result variable] Return value of GET SETPOINT function block

Return Value:

Byte 0

Latched	Bit0 0 = no, 1 = yes
Feeding	Bit1 0 = no, 1 = In Progress
Fast Feeding	Bit2 0 = no, 1 = In Progress
Low Tolerance Weight	Bit3 0 = Above -Tol Value 1 = Under –Tol Value
High Tolerance Weight	Bit4 0 = Below +Tol Value 1 = Over +Tol Value
In Tolerance	Bit5 0 = Out of Tolerance 1 = In Tolerance
Reserved	Bit6
Reserved	Bit7

Byte 1

Reserved	Bit8
Pause	Bit9 1 = Pause State
Reserved	Bit10

- Note: The Tolerance bits are only returned when the IND500x Target Mode is set to Over/Under.

5.10. Get Weight

Scale Get Weight

Get the legal-for-trade weight and status for the selected scale and node.

5.10.1. Properties

Scale	[Integer] Selects the scale. Legal values are 1.
Node	[Integer] Selects the node in the cluster. Legal values are 0. 0 refers to the local node.
Gross	[String or Double variable] Gross weight of the selected scale. The data type depends on the data type of the variable. If the data type is STRING, GETWT returns a string representation. If the data type is DOUBLE, GETWT returns a floating point representation of the gross weight.
Net	[String or Double variable, Output/Result variable] Net weight of the selected scale. The data type depends on the data type of the variable. If the data type is STRING, GETWT returns a string representation. If the data type is DOUBLE, GETWT returns a floating point representation of the net weight.
Tare	[String or Double variable, Output/Result variable] Tare weight of the selected scale. The data type depends on the data type of the variable. If the data type is STRING, GETWT returns a string representation. If the data type is DOUBLE, GETWT returns a floating point representation of the tare weight.
Scale Status	[Integer variable, Output/Result variable] Scale status such as Motion, Data OK, Zero, and Range status of the selected scale.

Bit 6	1 = Data OK	Bit 3-2	00 = single range
Bit 5	1 = Scale in Motion		01 = weight range 1
Bit 4	1 = Center of Zero		02 = weight range 2
			03 = weight range 3
		Bit 1	1 = Net Mode
		Bit 0	1= Preset Tare

Units	[String or Integer variable, Output/Result variable] Current units of the scale. The data type depends on the data type of the variable. If the data type is STRING, GETWT returns a string representation. If the data type is INTEGER, GETWT returns an integer representation of the tare weight.

0 = None	5 = ton (tons)
1 = lb (pounds)	6 = lb&oz (pound ounces)
2 = kg (kilograms)	7 = ozt (troy ounces)
3 = g (grams)	8 = dwt (penny weight)
4 = t (metric tons)	9 = oz (ounces)
	10 = custom

5.11. Setpoint Setup

Scale Setpoint Setup

Setup the control values for a setpoint. Start the setpoint running. The setpoint turns on the feed-control signals to initiate the feed. It compares the setpoint target value with the dynamic weight value at a rate of up to 50 comparisons per second. It shuts off the feed when system reaches the target conditions defined in the setpoint data.

5.11.1. Properties

Scale	[Integer] Selects the scale. Legal values are 1.
Node	[Integer, Optional] Selects the node in the cluster. Legal values are 0. 0 or a non-existent parameter refers to the local node.
Setpoint	[Integer] Number of the setpoint that this function initializes. Legal values are: 1
Source	Dynamic value that the setpoint compares to the coincidence value. Options: Net Weight, Gross Weight, Rate, Count
Action	Transfer action that the setpoint uses to transfer material during the feed. Options: 1-speed fill, 2-speed fill, 1-speed discharge, 2-speed discharge, Dump to empty, Classify – no motion check, Classify – motion check
Target Weight	[Double] Target value for the setpoint. The setpoint compares the dynamic source% value with the target value and shuts off the setpoint when they coincide. All weight units are in the primary weight units for the scale.
Latching	Disables the latching mechanism for the setpoint. When latching is enabled, the setpoint never automatically re-enables a feed-control once it has disabled the feed. Reasons for disabling a feed-control include error conditions encountered or target reached. The operator or an application program must take action to re-enable the setpoint feed. Options: Disable, Enable (default)
Visualization	Always 0.
Name	[String] Defines the name of the setpoint. The terminal displays the name in the SmartTrac display.
Spill Value	[Double] Defines the weight-value prior to cutoff that the setpoint will shut-off the feed control. The setpoint shuts of the feed-control when it encounters the target weight# – spill value#. This compensates for the filtering delays and delays in closing the mechanical valve.
Fine Feed Value	[Double] Defines switch-over weight when using two-speed feed control. The setpoint shuts off the high-speed-feed-control when it encounters the target weight# - spill value# - fine feed value#.
Upper Tolerance	[Double] Defines the allowed tolerance weight above the target weight# for a successful feed.
Lower Tolerance	[Double] Defines the allowed tolerance weight below the target weight# for a successful feed.
SetPoint Status	[Integer variable, Output/Result variable] Return value of SetPoint function block. Return Status:

```
SETPOINT_STATUS
SETPOINT_SETUP_COMPLETED_SUCCESSFULLY    0
SETPOINT_SETUP_IN_PROGRESS               1
SETPOINT_NOT_ACTIVE                      2
INVALID_SETPOINT_FUNCTION_PARAMETER     98
CANNOT_ACCESS_SETPOINT_SD                        99
```

5.12. Setpoint Target

Scale Setpoint Target

Set a new coincidence target for this setpoint. An application can use this command to adjust a setpoint while it is in progress.

5.12.1. Properties

Setpoint	[Integer] The number of the setpoint that this function initializes. Legal values are: 1
Node	[Integer, Optional] Selects the node in the cluster. Legal values are 0. 0 or non-existent parameter refers to the local node.
Target Weight	[Double] Target value for the setpoint. The setpoint compares the dynamic source% value with the target value and shuts off the setpoint when they coincide. All weight units are in the primary weight units for the scale
Setpoint Status	[Integer variable, Output/Result variable] The output status for the operation. Please refer to the status definition under the Setpoint Setup function block.

5.13. Set Units

Scale Set Units

Select the primary or secondary units operation for a scale.

5.13.1. Properties

Scale	[Integer] Selects the scale. Legal values are 1.
Node	[Integer, Optional] Selects the node in the cluster. Legal values are 0. 0 or a non-existent parameter refers to the local node.
UnitsType	Selects the type of units switching. Options: Primary, Secondary, Switch to alternate units.
Basic Function Status	[Integer variable, Output/Result variable, Optional] Return value of this function block. Return Status:

```
BASIC_FUNCTION_STATUS
FUNCTION_COMPLETED_SUCCESSFULLY      0
FUNCTION_IN_PROGRESS                 1
FUNCTION_NOT_ENABLED                 2
FUNCTION_NODE_NOT_ONLINE             3
INVALID_BASIC_FUNCTION_PARAMETER    98
CANNOT_ACCESS_FUNCTION_SD_TRIGGER   99
```

5.14. Stop Setpoint

Scale Stop Setpoint

Stop the current feed for the setpoint.

5.14.1. Properties

Setpoint	[Integer] The number of the setpoint that this function initializes. Legal values are 1.
Node	[Integer, Optional] Selects the node in the cluster. Legal values are 0. 0 or non-existent parameter refers to the local node.
Status	[Integer variable, Output/Result variable, Optional] The output status for the operation. Please refer to the status definition under the Setpoint Setup function block.

5.15. Switch Resolution

Switch the resolution of the scale weight between its standard resolution and its ″x10″ resolution. The ″x10″ resolution provides an additional decimal position of weight resolution in the weight calculation.

5.15.1. Properties

Scale	[Integer] Selects the scale. Legal values are 1.
Node	[Integer] Selects the node in the cluster. Legal values are 0. 0 refers to the local node.
Status	[Integer variable, Output/Result variable, Optional] The output status for the operation. Please refer to the status definition under the Set Units function block.

5.16. Tare

Tare scale. If the function call contains a tare value, it will be used. Otherwise, perform a pushbutton tare.

5.16.1. Properties

Scale	[Integer] Selects the scale. Legal values are: 1.
Node	[Integer, Optional] Selects the node in the cluster. Legal values are: 0. 0 or non-existent parameter refers to the local node.
Tare Value	[Double, Optional] Pre-defined tare value.
Status	[Integer variable, Output/Result variable, Optional] The output status for the operation. Please refer to the status definition under the Clear Tare function block.

5.17. WTSTR$

Convert floating-point weight from a floating-point representation to a string representation. Calculate the rounded the weight according to the scale increment-size settings for a particular scale.

5.17.1. Properties

Weight	[Float] The floating-point weight value to convert.
Scale	[Integer] Selects the scale. Legal values are 1. The output string will be formatted appropriately for the scales increment size.
Units	[Optional] The weight units of the weight value. Options: None, pounds, kilograms, grams, metric Tons, tons, oz. Units% is an optional value. When the units are not specified, WTSTR$ uses the primary units of the selected scale.
Converted Weight	[String variable, Output/Result variable] The formatted output string.

5.18. Zero Scale

Attempt to Zero the specified scale.

5.18.1. Properties

Scale	[Integer] Selects the scale. Legal values are 1.
Node	[Integer, Optional] Selects the node in the cluster. Legal values are 0. 0 or non-existent parameter refers to the local node.
Tare Status	[Integer variable, Output/Result variable, Optional] Return status of this function block.

Return Status

```
ZERO_STATUS
ZERO_COMPLETED_SUCCESSFULLY              0
ZEROING_IN_PROGRESS                      1
SCALE_IN_MOTION_DURING_ZERO              2
ILLEGAL_SCALE_MODE_DURING_ZERO           3
SCALE_OUT_OF_ZEROING_RANGE               4
INVALID_ZERO_FUNCTION_PARAMETER         98
CANNOT_ACCESS_ZERO_SD_TRIGGER           99
```

6 Interface Commands

6.1. Command Reference

6.1.1. File Commands

Command	Usage
Close	Closes an open file or serial port.
Input	Reads input from the keyboard, serial port, or a sequential file.

6.1.2. TCP/IP Commands

Command	Usage
Accept	Allows the TaskExpert application to accept new connection requests that remote clients are initiating.
Close Socket	Allows the TaskExpert application to close an established TCP/IP socket connection.
Connect	Initiates a Client connection to a remote Server.
GetIP	Gets the IP Address of local node or node in the cluster.
Listen	Initializes the TCP/IP Server to begin queuing the connection requests for the host port.
Receive	Allows TaskExpert to receive data over an established Client or Server connection.
RecvArray	Allows the application to receive data.
Send	Sends data over an established Client or Server connection.
SendArray	Allows the application to send array data as bytes.
Socket	Creates a socket for a subsequent use in a CONNECT command.

6.1.3. Analog Output Commands

Command	Usage
Set Analog Output Zero	Allows zero output value to be written to the Analog Output channel.
Set Analog Output Span	Allows span output value to be written to the Analog Output channel.
Set Analog Output Value	Sets output values for the Analog Output value.

6.1.4. Serial Commands

Command	Usage
Flush	Discards received data in the BIOS serial input buffer.
Open COM	Prepares a serial port for use as a file device.
Print	Outputs the data to the specified serial port.
Width	Assigns an output line width to the LPRINT device, serial port, or a file.
Width In	Permits dynamic reassignment of the maximum serial input length.

6.1.5. Discrete I/O Commands

Command	Usage
Pulse Discrete I/O	Turns a discrete output on for a specified number of milliseconds and then off.
ReadDI	Reads the static value of a discrete input.
Set Discrete IO Option	Turns a discrete output on or off.

6.1.6. Print Commands

Command	Usage
Custom Print	Issues the Custom Print command.
Demand Print	Issues the Demand Print command.
Set Template	Sets a Print Template in Shared Data.

6.1.7. Connection Commands

Command	Usage
Define Connection	Defines an input or output connection in the Data Connections table in Shared Data.

6.2. Accept

Interfaces Accept

ACCEPT allows the TaskExpert application to accept new connection requests that remote clients are initiating. If ACCEPT finds a new connection, it creates a new socket for the new connection.

6.2.1. Properties

Time Out	[Integer] TaskExpert application must supply an integer number that is the timeout value in milliseconds for the ACCEPT. If the timeout value is 0, ACCEPT waits indefinitely. Otherwise, ACCEPT waits for a new connection for the timeout period. If there is no new connection, the TaskExpert application must periodically issue the ACCEPT command to know when a new connection occurs.

New Socket	[Integer variable, Output/Result variable] If successful, the ACCEPT returns a positive number that is the socket number of the new connection. If there is no new connection, ACCEPT returns a negative number that is the failure status. Please refer to TCP FUNCTION STATUS (LISTEN). When there is a successful new connection, ACCEPT also sets the IP address of the remote node that initiated the connection in the variable IP.

6.3. Close

Interfaces Close

Closes an open file or serial port. Use CLOSE after all input and output operations for a file or devices are concluded. CLOSE releases the memory space reserved in the buffer for the open file or serial port.

6.3.1. Properties

File/Serial IO Number	File number in "#<Filenum>" format.

- Note: Each open file must have its own CLOSE command. When writing a file, you should frequently close the file to avoid losing data in the event of a power failure.

6.4. Connect

Interfaces Connect

The CONNECT function initiates a Client connection to a remote Server. The remote Server must be listening with an ACCEPT command for the CONNECT to succeed.

6.4.1. Properties

Socket	[Integer] Socket number to connect.
IP Address	[String] IP Address.
Host Port	[Integer] Port number within the host.
Status	[Integer variable, Output/Result variable] Return value is an integer variable that is the status of the connection. Please refer to TCP FUNCTION STATUS (LISTEN).

6.5. Custom Print

Interfaces Custom Print

Issue the Custom Print command for the specified custom print number and node.

6.5.1. Properties

Print Number	[Integer] Selects the custom print number. Legal values are 1 to 10.

Node	[Integer, Optional] Selects the node in the cluster. Legal values are 0. 0 or a non-existent node parameter refers to the local node.
Printing Status	[Integer variable, Output/Result variable, Optional] Return value of this function block. Please refer to the PRINTING STATUS (DEMAND) statuses.

6.6. Demand Print

Issue the Demand Print command for the selected scale. Demand Print is a transaction print that may record the transaction.

6.6.1. Properties

Scale — [Integer] Selects the scale. Legal values are 1.

Node — [Integer] Selects the node in the cluster. Legal values are 0. 0 or non-existent parameter refers to the local node.

DemandPrint Status — [Integer variable, Output/Result variable, Optional] Return value of function block.

Return status:

PRINTING_STATUS	
PRINTING_COMPLETED_SUCCESSFULLY	0
PRINTING_IN_PROGRESS	1
PRINTING_CONNECTION_NOT_FOUND	2
PRINTING_BUSY	3
PRINTING_ERROR	4
PRINTING_NOT_READY_TO_PRINT	5
PRINTING_SCALE_IN_MOTION	6
PRINTING_SCALE_OVERCAPACITY	7
PRINTING_SCALE_UNDER_ZERO	8
INVALID_PRINT_FUNCTION_PARAMETER	98
CANNOT_ACCESS_PRINT_SD_TRIGGER	99

6.7. Get IP

Get IP Address of local node or node in the cluster.

6.7.1. Properties

Node — [Integer, Optional] If node = 0 or node is not specified, return the IP address of the local node. Otherwise, return the IP address of the node in the cluster.

IP Address — [String variable, Output/Result variable] IP Address string. If successful, IP Address string. Otherwise, NULL string.

6.8. Input

Interfaces Input

Reads input from the keyboard, serial port, or a sequential file. When reading a sequential file, the file must be "comma-delimited." That is, commas between items and quotation marks around the strings in the file are required.

6.8.1. Properties

File/Serial IO Number	File number in "#<Filenum>" format.
Value	List of one or more variables to set data. Items must be separated by commas.
InputType	Normal, Line Select Normal if the data has to be stored in multiples. Select Line if a single line input is needed. The Line input type sequentially reads all characters of an entire line without delimiters from a sequential file up to the next carriage return into string variable.

- Note: Function must use a TaskGlobal for the Value parameter variable.

6.9. Listen

Interfaces Listen

LISTEN function initializes the TCP/IP Server to begin queuing the connection requests for the host port. Subsequently, the ACCEPT command allows the TaskExpert application to begin accepting the connection requests from a remote node. Remote clients initiate the connection requests with the CONNECT command.

6.9.1. Properties

Host Port [Integer] Host port number.

Status [Integer variable, Output/Result variable, Optional] Return status of this function block.

```
TCP_FUNCTION_STATUS
TCP_FUNCTION_COMPLETED_SUCCESSFULLY      0 (listen socket
     number)
TCP_LISTEN_ALREADY_ACTIVE               -1
TOO_MANY_SOCKETS                        -2
TCP_GETADDRINFO_FAILED                  -3
TCP_SELECT_FAILED                       -4
TCP_ACCEPT_FAILED                       -5
TCP_INVALID_SOCKET                      -6
TCP_SEND_ERROR                          -7
TCP_RECEIVE_TOO_LONG                    -8
TCP_CONNECT_FAILED                      -9
TCP_SOCKET_DISCONNECTED                 -10
TCP_RECEIVE_ERROR                       -11
TCP_ACCEPT_TIMEOUT                      -12
TCP_PROGRAM_TRIGGER_RECEIVED            -13
INVALID_TCP_FUNCTION_PARAMETER          -98
CANNOT_ACCESS_TCP_SD_TRIGGER            -99
```

6.10. Print

PRINT outputs the data to the specified serial port.

6.10.1. Properties

IO Number	[Optional] File number in "#<Filenum>" format. Value is valid only if the 'PrintType' is 'Normal'. Otherwise, it is ignored.
Style	Print normal or formatted data.
Format	[String, Optional] Format of the output data.
Value	List of one or more numeric or string expressions to print. Items must be separated by commas or semicolons. ■ Note: The absence of a semicolon (;) at the end of the line means to insert a new line (LF).
PrintType	Select Print type. Options: PRINT(normal), LPRINT(line), TPRINT(terminal).

■ Note: When using a comma to separate the variables or text strings, a TAB will be added between strings. A semicolon may be used instead to eliminate any space between strings.

6.11. Define a Connection

Define anew input or output connection in the Data Connections table in Share Data. If there is already a duplicate entry, do not insert the new entry but return a successful status anyway. This command only writes to the local Shared Data

6.11.1. Properties

Input/Output Type	**Input**: [Input] Type of input connection Options: None, Scale Commands (CTPZ-style), Scale Commands (SICS remove levels 0 & 1), Bar codes **Output**: [Integer] Type of output connection None, Scale transaction and custom demand print, Continuous output, Continuous demand print, Reports, Action log output, Short continuous, Totals reports
Input Properties	
Template	[Integer] Applicable only to bar code connections Options: None, Programmable bar code template 1, Programmable bar code template 2
Port	[Integer] IO port number. There is only one IO port per input data connection. 1-6 are Serial Ports 1-6 (IND500x)
Status	[Integer variable, Output/Result variable, Optional] Return status of this function block. Please refer to the PRINTING STATUS (DEMAND) statuses.

Output Properties

Trigger	[Integer] Device or function that triggers the output Options: None, Scale 1, Scale 2, Custom Print 1-3
Template	[Integer] Print Template associated with te Data Connection Options: Standard template, Programmable templates 1-5
Port1	[Integer] IO port for the output connection
Port2	[Integer] IO port for the output connection
Port3	[Integer] IO port for the output connection
Port4	1-6 are Serial Ports 1-6.
Port5 Port6	14-15 are TCP/IP Demand Print Message streams 2-3 for remote data connections. A client application must connect to the Shared Data Server to receive data from this output connection 16 is TCP/IP message stream for continous output
Status	Return status of this function block. Please refer to the PRINTING STATUS (DEMAND) statuses.

6.12. Pulse Discrete IO

Interfaces Pulse Discrete IO

Turn a discrete output on for a specified number of milliseconds and then off. You identify a Discrete IO by the node, the local board slot or remote IO slot, and position within that slot.

6.12.1. Properties

Node	[Integer, Optional] Selects the node in the cluster. Legal values are 0. 0 or non-existent parameter refers to the local node.
Slot	[Integer] There are 2 legal local board slots: 1, 2 There are 3 legal remote IO slots: 3 – 5 These define the local board slot or remote IO slot that contains the Discrete IO.
Discrete Output	[Integer] The discrete output’s position within the slot. Legal positions for the local board slot 1 are 1 – 3. Legal positions for the local board slot 2 are 1 – 8. Legal positions for the remote IO are: 1 – 6.
Duration	[Integer] Number of milliseconds to turn the discrete output on before turning it off.
Status	[Integer variable, Output/Result variable] Return value of this function block. Please refer to BASIC FUNCTION STATUS.

6.13. Read Array

Reads a shared data byte array (Aby), Boolean array (ABI) or Long array (AL) into an arrow.

6.13.1. Properties

Shared Data name	Name of shared data that is an array type, or byte, bool or long
Array name	Name of the array where the shared data array data will be stored.
Offset%	Optional array index location into which the data should be placed. If Offset% is not defined, data wil be placed at index 0.
Status	[Integer variable, Output/Result variable] Return value of this function block. Please refer to BASIC FUNCTION STATUS.

6.14. Open COM

Prepares a serial port for use as a file device. You can access a serial port that has been set up as a demand print serial connection or a custom print serial connection. You cannot access a serial port from TaskExpert if it has been set up as a continuous output connection or as an input connection. If the serial port is on the local terminal, it can be accessed even if it is not set up in a connection.

6.14.1. Properties

Device Number	Serial number in "#<Device Number>" format. Valid Device Numbers are #0 to #7.
Port	[Integer] Port number which specifies the serial port to be used for communications. Legal values are 1 to 3.
Timeout	[Integer] Specifies the time-out value to wait for a serial input message in decimal milliseconds. The default value is zero milliseconds, or no time-out value. The maximum time-out value is 30,000 milliseconds.
Length	[Integer, Optional] Specifies the maximum input length for a serial input message. The default length is 80 bytes.
Terminating Character	[Integer, Optional] Specifies an optional terminating character for the serial input message. Its value is specified in decimal. When the input command encounters the terminating character, it returns the characters up to and including the terminal character in the serial message as a string variable.
CR	Specifies that a carriage return character is to be inserted at the end of any serial output message.
Event	Allocates an event which my trigger an event processing routine when a serial input operation completes.

Express Print	Selects the ″express print″ option. Normally, PRINT data is sent to the serial port when either a ″new line″ character is encountered or the print data length exceeds the WIDTH value. Using Express Print causes PRINT data to be sent to the serial port immediately at completion of the PRINT statement, even when there is no terminating ″new line″ character.
NULL Output	Enables the inputting and outputting of NULL (0) characters through the serial I/O. Since the NULL character is a terminator for strings, you must send and receive a special sequence of characters for the NULL character. The sequence ″DLE 0xff″ represents the NULL character in an application. The sequence ″DLE DLE″ represents a single DLE character. The following string represents a null character: null$ = chr$(16) + chr$(255).
Mode	The mode can be either input or output. No matter which is chosen, you can do input or output to the specified serial device.

6.15. ReadDI

Read the static value of a discrete input. To get an event trigger when a discrete input changes state, you must use a DEFSHR EVENT statement.

6.15.1. Properties

Node	[Integer, Optional] Selects the node in the cluster. Legal values are 0. 0 or non-existent parameter refers to the local node.
Slot	[Integer] Local board slot or remote IO slot that contains the Discrete IO. There are 2 legal board local slots: 1, 2 There are 3 legal reote IO slots: 3 – 5
Discrete Input	[Integer] Discrete input's position within the slot. Legal positions for the local board slot 1 are 1 – 3. Legal positions for the local board slot 2 are 1 – 8. Legal positions for the remote IO are: 3 – 5.
Status	[Integer variable, Output/Result variable, Optional] Return value of this function block. For error statuses, please refer to BASIC FUNCTION STATUS. Return value: 0 = Discrete input OFF 1 = Discrete input ON

6.16. Receive

Receive command allows the TaskExpert to receive data over an established Client or Server connection.

6.16.1. Properties

Socket	[Integer] Socket to connect.
Length	[Integer] Length of the receiving string. The maximum received data length on each call is the TaskExpert maximum string size (1000 bytes).
Time Out	[Integer, Optional] Duration to wait for incoming data in milliseconds. If the timeout value is zero, RECEIVE will wait indefinitely for the incoming data. Otherwise, RECEIVE returns back the data as soon as it is available or returns a NULL data string after the timeout. After a timeout, the TaskExpert application must periodically re-issue the RECEIVE command to see if there is more data.
Input String	[String variable, Output/Result variable] Returns the string that was received through the socket. If it is successful, RECEIVE returns the data string. If there is no data available on the connection or an error in the RECEIVE command, RECEIVE returns the NULL string. RECEIVE generates a BASIC error or prints an error message to the LPRINT connection, depending on the severity and type of the error.

6.17. RecvArray

RecvArray command allows the application to receive data. For TCP sockets the data is from an established Client or Server connection.

6.17.1. Properties

Socket	[Integer] Socket number
Array Name	[String] Name of the integer array to store the received bytes. The maximum size is the length of the dimensioned array.
Time Out	[Integer] Length of time in milliseconds that RecvArray will wait for incoming data. If the timeout value is zero, RecvArray will wait indefinitely for the incoming data. Otherwise, RecvArray returns back the data as soon as it is available or returns zero No of Chars after the timeout. After a timeout, the Task Expert application must periodically re-issue the RecvArray command to see if there is more data.
No of Chars	[Integer variable, Output/Result variable] The return value is an integer variable. If RecvArray is successful, it returns a positive number that is the number of characters received. If it fails, RecvArray returns a negative number that is the failure status. Please refer to TCP_FUNCTION_STATUS.

6.18. Send

The SEND command allows TaskExpert to send data over an established Client or Server connection.

6.18.1. Properties

Socket	[Integer] Socket number
String to send	[String] String to be sent
No of Chars	[Integer variable, Output/Result variable] Returns the number of chars sent through the socket. If SEND is successful, it returns a positive number that is the number of characters sent. If it fails, SEND returns a negative number that is the failure status. Please refer to TCP FUNCTION STATUS (LISTEN).

6.19. SendArray

The SendArray command allows the application to send array data as bytes. The lower byte (0-255) of each array element is sent. For TCP sockets it is sent over an established Client or Server connection.

6.19.1. Properties

Socket	[Integer] Socket number
Array Name	[String] Name of the integer array containing the bytes to send
Length	[Integer] Number of array elements to send. If this is not specified all elements will be sent.
No of Chars	[Integer variable, Output/Result variable] The return value is an integer variable. If SendArray is successful, it returns a positive number that is the number of characters sent. If it fails, SendArray returns a negative number that is the failure status. Please refer to TCP_FUNCTION_STATUS.

6.20. Set Analog Output Zero

Put the Analog Output channel in "Application" mode so that a Task Expert application can write output values to the Analog Output channel. Set the Preset Zero value of the output range for the channel. The zero can be manually trimmed in Setup.

6.20.1. Properties

Channel	[Integer] Analog Output channel number. Legal values are 1.

Value	[String] Specifies Preset Zero value.

Status:

```
Success        0
ERR_INVALID_CHANNEL       -1
```

6.21. Set Analog Output Span

Put the Analog Output channel in "Application" mode so that a Task Expert application can write output values to the Analog Output channel. Set the Preset Span value of the output range for the channel. The span can be manually trimmed in Setup.

6.21.1. Properties

Channel	[Integer] Analog Output channel number. Legal values are 1.
Value	[String] Specifies Preset Span value.

Status:

```
Success        0
ERR_INVALID_CHANNEL       -1
```

6.22. Set Analog Output Value

After setting the Preset Zero and Preset Span values for the output range for the channel, output values to the Analog Output channel can be set.

6.22.1. Properties

Channel	[Integer] Analog Output channel number. Legal values are 1.
Value	[String] Specifies the output value that is written to the channel.
Discrete Bits	[Integer] Two Boolean bits, each having a value of 0 or 1. Bit 1 is the Error bit. If the Error bit = 1, the Analog Output discrete Error bit is set to ON. Bit 0 is the Data OK bit. If the Data OK bit = 1, the Analog Output discrete Data OK bit is set to ON

Status:

```
SUCCESS        0
ERR_INVALID_CHANNEL       -1
ERR_INVALID_MODE   -2
```

6.23. Set Template

Interfaces Set Template

Set a Print Template in Shared Data. A Print Template can be up to 1000 bytes long. You may use up to 5 strings to initialize the template. The Template function concatenates the strings before writing them to Shared Data. This command only writes to the local Shared Data.

6.23.1. Properties

Template	[Integer] Selects the template number. Legal values are 1 to 5.
String1	[String] The BASIC strings that make up the template.
String2	[String, Optional] The BASIC strings that make up the template.
String3	[String, Optional] The BASIC strings that make up the template.
String4	[String, Optional] The BASIC strings that make up the template.
String5	[String, Optional] The BASIC strings that make up the template.
Status	[Integer variable, Output/Result variable, Optional] Return status of this function block. Please refer to BASIC FUNCTION STATUS.

6.24. Socket

Interfaces Socket

The SOCKET function creates a TCP socket for a subsequent use in a CONNECT command, which initiates a connection to a remote host using this socket.

6.24.1. Properties

Socket	[Integer variable, Output/Result variable] The return value is an integer variable. If it is successful, SOCKET returns a positive number that is the socket number. If it fails, SOCKET returns a negative number that is the failure status. Please refer to TCP FUNCTION STATUS (LISTEN).

6.25. Write Array

Interfaces Write Array

Reads a TaskExpert array and copies it into a shared data array.

6.25.1. Properties

Array name	Name of the array containing data to be copied.
Shared data name	Name of the shared data that is an array type of byte, bool or long.

Offset%	Optional array index location from which the data should be placed. If Offset% is not defined, data wil be copied starting from index 0.

7 File Commands

7.1. Command Reference

Command	Usage
Close	Closes an open file or serial port.
Data	Specifies values to be read by READ statements
FileCopy	Copies a file locally.
FileMove	Moves a file locally.
Input	Reads input from the keyboard, serial port, or a sequential file.
Kill	Deletes either a file or an entire file path
Open File	Accesses a file
Print	Outputs data to the specified serial port or writes data to a sequential file
Read	Reads values from a DATA statement and assigns them to variables
Restore	Allows DATA statements to be reread from a specified line
Swap	Exchanges the value of two variables
Write	Outputs delimited data to the sequential file.

7.2. Close

Closes an open file or serial port. Use CLOSE after all input and output operations for a file or devices are concluded. CLOSE releases the memory space reserved in the buffer for the open file or serial port.

7.2.1. Properties

File/Serial IO Number	File number in "#<Filenum>" format.

- Note: Each open file must have its own CLOSE command. When writing a file, you should frequently close the file to avoid losing data in the event of a power failure.

7.3. Data

Specifies values to be read by READ statements. DATA statements contain lists of values separated by commas.

7.3.1. Properties

Constant Data	[String Constant Only] One or more numeric or string constants specifying the data to be read. String constants containing commas, colons, or leading or trailing spaces are enclosed in quotation marks(" ")

7.4. FileCopy

Copy a file locally.

7.4.1. Properties

ExistFilename	[String] The path and name of the file to be copied
NewFilename	[String] The path and name of the new file.
Status	[Integer variable, Output/Result variable, Optional] The return value is an integer variable that is the status of the function block.

Status

```
SUCCESS       0
FAIL   1
```

7.5. FileMove

Move a file locally.

7.5.1. Properties

ExistFilename	[String] The path and name of the file to be moved.
NewFilename	[String] The path and name of the new file.
Status	[Integer variable, Output/Result variable, Optional] The return value is an integer variable that is the status of the function block.

Status

```
SUCCESS       0
FAIL   1
```

7.6. Input

File Input

Reads input from the keyboard, serial port, or a sequential file. When reading a sequential file, the file must be "comma-delimited." That is, commas between items and quotation marks around the strings in the file are required.

7.6.1. Properties

File/Serial IO Number	File number in "#<Filenum>" format.
Value	List of one or more variables to set data. Items must be separated by commas.
InputType	Normal, Line Select Normal if the data has to be stored in multiples. Select Line if a single line input is needed. The Line input type sequentially reads all characters of an entire line without delimiters from a sequential file up to the next carriage return into string variable.

- Note: Function must use a TaskGlobal for the *Value* parameter variable.

7.7. Kill

File Kill

The TaskExpert KILL command deletes a file. It supports variable names as well as fixed program names as command arguments.

7.8. Open File

File Open File

Accesses a file.

Note: The IND500x only supports sequential files.

7.8.1. Properties

File Name	[String] Path + Name of the file to open.
Mode	[Optional] Sequential files are opened as INPUT, OUTPUT, or APPEND. Opening a sequential file for OUTPUT creates a new file. Opening a sequential file for APPEND adds new records to the end of an existing file. **Note**: APPEND creates a file if it doesn't already exist.
File Number	File number in "#<Filenum>" format. **Example**, accessing file on USB memory: `"MSFS:\myFileName.txt"`

7.9. Print

Outputs data to the specified serial port or writes data to a sequential file.

7.9.1. Properties

IO Number	[Optional] File number in "#<Filenum>" format. Value is valid only if the 'PrintType' is 'Normal. Otherwise, this is ignored.
Style	None, Formatted.
Format	[String, Optional] Format of the output data. # Digit position . Decimal point position ^ Prints in exponential format _ Space + Sign Other characters are printed as literal data in the output. Use these characters to format string expressions: ! Print corresponding characters of string \ \ Print first *n* characters of string, where *n* is the number of blanks between slashes.
Value	List of one or more numeric or string expressions to print. Items must be separated by commas or semicolons. The absence of a semi-colon at the end of the line means to insert a new line (LF).
PrintType	Normal, Line, Terminal.

- **Note**: When using a comma to separate the variables or text strings, a TAB will be added between strings. A semicolon may be used instead to eliminate any space between strings.

7.10. Read

Reads values from a DATA statement and assigns them to variables. Values are always read in the order in which they appear in the Data statements.

7.10.1. Properties

Variable	List of one or more variables. Items must be separated by commas.

7.11. Restore

Allows DATA statements to be reread from a specified line. Enables a program to read data selectively based on a particular condition.

7.11.1. Properties

DATA statement	[Optional] If the statement is omitted, the next READ accesses the first item in the first DATA statement.

7.12. Swap

Exchanges the values of two variables.

7.13. Write

Outputs delimited data to the sequential file. The WRITE function inserts commas between items and quotation marks around strings as they are written. The WRITE function writes values in a form that can be read into separate variables by the INPUT statement.

7.13.1. Properties

File/Serial IO Number	File number in "#<Filenum>" format.
Value	List of one or more numeric or string expressions to write. Items must be separated by commas or semicolons.

8 Standard Table Commands

8.1. Command Reference

Command	Usage
Create Standard Tables	Creates ten new tables in their defined format, together with an index for each.
Open Tables	Opens the currently existing Standard Tables.
Close Standard Tables	Terminates access to Standard Tables.
Clear or Delete Standard Tables	Clears or deletes the current standard tables.
Get Row	Retrieves rows from Standard Tables, one at a time.
Set Row	Inserts a new row into a specific table.
Select Row	Selects a single row, or multiple rows, from a table.
Next Row	Selects the next row from a table, after Select Row selects the first.
Set Item	Sets the value of an item in selected row/s of a table.
Add Item	Adds a value to an item in selected row/s of a table.
Delete Row	Deletes one or more rows from a table.
Select	Selects all rows or specific rows by any record field from a table.

When using A2 table , the share data ds0111 must be set first. Programming by Direct Code is supported.

For example, user can use "ds0111@ = 2" to set the target table mode as Over Under.

1 = Basic Auto Filling, 2 = Over Under, 4 = Manual Filling

A1 - Tare Table

DB Field	Name	Description
ID		
shortID	ID	Tare ID
description	Description	Tare Description
data1	Tare	Tare Value

data2	Unit	Weighing Unit
data3	nTol	Negative Tolerance
data4	pTol	Positive Tolerance
data5	N	Total Count
data6	Total	Total Weight

A2 - Target Table

DB Field	Name	Description
ID		
shortID	ID	Target ID
description	Description	Target Description
data1	Target	Target value
data2	Unit	Target Units
data3	Source	Source
data4	nTol	Negative Tolerance
data5	pTol	Positive Tolerance
data6	TolType	Tolerance Type
data7	N	N
data8	Total	Total
data9	TotalType_En	Total Type Enable
data10	Spill	Spill
data11	Feed	Feed

8.2. Create Standard Database Tables

Standard Tables Create Standard Table

Create a new Standard Table. This command creates the ten data tables A0 – A9 in their defined format, and creates indexes for the entry # and description columns. The Standard Table file resides in the Flash.

Each table has 15 columns with the following column names and default formats:

```
ID           INT IDENTITY (1,1) NOT NULL PRIMARY KEY
shortID      NVARCHAR(16)
description  NVARCHAR(40)
data1        NVARCHAR(16)
data2        NVARCHAR(16)
data3        NVARCHAR(16)
data4        NVARCHAR(16)
data5        NVARCHAR(16)
data6        NVARCHAR(16)
data7        NVARCHAR(16)
data8        NVARCHAR(16)
data9        NVARCHAR(40)
data10       NVARCHAR(40)
data11       NVARCHAR(40)
data12       NVARCHAR(40)
```

Create Standard Tables generates an index of both the entry number ID and the description columns for fast lookups of rows using these index columns as keys.

TaskExpert has several "Table" functions that operate only on the Standard Tables.

8.2.1. Properties

Standard Table	Tables A0 – A9.
TableName	Table name, such as CUSTOMER, PRODUCT, or SETPOINT Note: Names cannot include spaces.
ShortIDName	The name of short id.
Description	The name of description. [Note: This cannot be an empty string.]
Data1 Name	The name of first data.
Data1 Type	The data type of first data, such as: integer type :% float type : # character type: $ Default is $.

8.3. Open Standard Database Tables

Standard Tables Open Tables

Open the currently existing Standard Tables for access within TaskExpert.

Note: The A1 and A2 tables are reserved for the terminal's Tare and Target tables.

8.3.1. Properties

Standard Table	Tables A0 – A9.

8.4. Close Standard Database Tables

Standard Tables Close Standard Table

Terminate access to Standard Tables.

8.5. Clear or Delete Standard Database Tables

Standard Tables Clear/Delete Standard Table

Clear or Delete standard tables.

8.5.1. Properties

Standard Table	Tables A0 – A9.
Action	Clear or Delete

8.6. Get Row

Standard Tables Get Row

After a SELECTROW or SELECT command selects a "rowset" from the Standard Tables, the GETROW command enables the TaskExpert application to retrieve the rows one at a time from the rowset until it retrieves all rows. A rowset is a set of one or more rows.

```
    GETROW numCol%, column1, column2,... columnN
```

The `numCol%` parameter gives the number of columns in the row retrieved by the `GETROW` command. If `numCoL%` is 0, the end of the rowset is reached and there are no more rows to retrieve.

The `GETROW` command copies data from the retrieved row into variables `column1` through `columnN.` The type of the `GETROW` column variables should match the data type of the columns in the rowset. However, TaskExpert attempts to do a data type conversion if the types are different. If the number of variables does not match the number of columns in the row, TaskExpert copies data up to the lesser of these two. It returns the number of columns copied in the `numCoL%` `variable.`

8.7. Set Row

Standard Tables Set Row

Insert a new row entry into a specific table in the Standard Tables.

8.7.1. Calling arguments

Table	Tables A0 – A9.
Short ID	[String] Contents of short ID field.
Description	[String] Contents of the description column in the row.
Data1	Contents of data column 1.
Data 2 – 12	Contents of the identified data column (optional).
Return values	`0 = SUCCESS, 1 = FAILED`

The `SETROW` command automatically converts the parameter data to string data, if necessary, before submitting it to the Standard Table.

The Standard Table automatically generates an entry number for the new row, using the next available number in sequence. Since the entry number is the Primary Key for the table, this guarantees that the entry number will be unique for each row. If an existing entry is deleted, that entry number becomes unused and the Standard Table can reuse it. For example: If three records are added, their entry numbers will be 1,2,3. If DELROW(2) is then used to delete the second row, and another record is subsequently added, the entry number of the new record will be 2. When the next new record is added, its entry number will be 4.

8.8. Select Row

Standard Tables Select Row

Select all rows or specific rows by description or by entry number from a table in the Standard Tables. The `SELECTROW` function also returns the first selected row from the table. Use the `NEXTROW` Table function or the `GETROW` command to retrieve the subsequent rows.

8.8.1. Short ID/Entry Number Usage

String variable	Search based on ShortID field.
Integer	Search based on unique Entry Number.
Empty	Returns all records in the selected table.

8.8.2. Calling Arguments

Table	Tables A0 – A9.

Short ID	[String] ShortID column for the selected row(s). The shortID column is not necessarily unique for each row so this function can select multiple rows.
Entry Number	[Integer] Entry number for the selected row. This number is unique for each row so this identifier will return at most one row.
Return Value	Number of columns. This function returns the number of columns successfully retrieved in the first selected row. If the return value is zero, no row is selected.

8.8.3. TaskExpert Variables

The `SELECTROW` function sets the column values for the first selected row in the following TaskExpert variables. The TaskExpert application can retrieve row data from these variables. If the data in the table contains less than the full 15 columns, the TaskExpert Interpreter sets the unused variables to a null value.

entryNumber%	Column 1
shortID$	Column 2
description$	Column 3
data1 - data12$	Columns 4 to 15

8.8.4. Shared Data Fields

The SELECTROW function also sets the column values for the first selected row in the following Shared Data variables. "--" represents the instance number for the table. A0 uses instance "01"; A1 uses instance "02"; A2 uses instance "03"; A3 uses instance "04"; A9 uses instance "10", etc. You can build these Shared Data fields into a Print Template so that the IND500x can automatically insert the table values into a print ticket or report.

dd--01 to dd--15	Columns 1 to 15

8.9. Next Row

Retrieve the next row from a rowset from the Standard Tables. The `SELECTROW` function returns the first selected row from the table. Use the `NEXTROW` table function to retrieve subsequent rows.

The return values, TaskExpert variables, and Shared Data fields are the same as for the `SELECTROW` function.

8.9.1. Properties

Number of Columns	[Integer variable, Output/Result variable] Returns the number of columns successfully retrieved in the first selected row. If the value is zero, no row is selected.

8.10. Set Item

Standard Tables Set Item

Set the value of an item in a selected row(s) in a Standard Table. When multiple rows are selected, `SETITEM` writes the item to all selected rows.

8.10.1. Calling Arguments

Table	Tables A0 – A9.
ShortID	[String] ShortID column for the selected row(s). The shortID column is not necessarily unique for each row so this function can select multiple rows. If the shortID selects multiple rows, the Standard Table modifies the column value in all selected rows.
entryNumber	[Integer] Entry number for the selected row. This number is unique for each row so this function will select at most one row.
Item	[Integer] Data field 1 – 12 in selected row(s) to be modified. 0 = Description field.
Data	The data value to be inserted into the selected row-column item.
Return Value	`0 = SUCCESS, 1 = FAILED`

8.11. Add Item

Standard Tables Add Item

Add the value to an item in a selected row(s) in a table of the Standard Tables. When multiple rows are selected, `ADDITEM` adds the result to the first row-column item and writes the item result back to all selected rows.

8.11.1. Calling Arguments

Table	Tables A0 – A9.
ShortID	[String] ShortID column for the selected row(s). The shortID column is not necessarily unique for each row so this function can select multiple rows. If the shortID selects multiple rows, the Standard Table modifies the column value in all selected rows.
entryNumber	[Integer] Entry number for the selected row. This number is unique for each row so this function will select at most one row.
Item	[Integer] Data field 1 – 12 in selected row(s) to be modified.
Data	The data value to be added to the selected row-column item.
Return Value	`0 = SUCCESS, 1 = FAILED`

8.12. Delete Row

Standard Tables Delete Row

Delete specific rows by shortID or by entry number from a table in the Standard Tables. When you delete a row, the Standard Table can reuse the entry number associated with the deleted row.

8.12.1. Calling Arguments

Table	Tables A0 – A9.
ShortID	[String] ShortID column for the selected row(s). The shortID column is not necessarily unique for each row so this function can select multiple rows. If the shortID selects multiple rows, the Standard Table modifies the column value in all selected rows.
entryNumber	[Integer] Entry number for the selected row. This number is unique for each row so this function will select at most one row.
Return Value	`0 = SUCCESS, 1 = FAILED`

8.13. Select

Standard Tables Select

Select all rows or specific rows by any record field from a table in the Standard Tables. Use the `GETROW` command to retrieve the first selected row. Use the `NEXTROW` Table function or the `GETROW` command to retrieve the subsequent rows. Or you can embed this command in DATAGRID command.

8.13.1. Parameters

Table	Tables A0 – A9.
SearchRecordName	[String] Searched record name (the name of fields), this name being generated in Create Standard Tables. For example :ID, Description.
WhereMatch	Match type – for example, <, <>, >, =, etc.
SearchRecordValue	Search match value; use the * to match all data.
OrderRecordName	Sort record name (the name of fields). This name is the same as SearchRecordName.

OrderType	`ASC = Return values in ascending order`
	`DESC = Return values in descending order`
	Default is ASC.

8.13.1.1. Example

Same as SQL command: SELECT * FROM A1 WHERE tare=32 ORDER BY ID:

```
Select 1,tare,=,32,id
```

Same as SQL command: SELECT * FROM A2 WHERE finefeed > 2 ORDER BY target:

```
Select 2,finefeed,>,2,target
```

8.13.1.1.1. Ascending Order (does not include argument)

Same as SQL command: SELECT * FROM A2 WHERE finefeed > 2 ORDER BY target. The values are returned in Ascending Order:

```
Select 2,finefeed,>,2,target
```

8.13.1.1.2. Ascending Order

Same as SQL command: SELECT * FROM A2 WHERE finefeed > 2 ORDER BY target ASC. The values are returned in Ascending Order:

```
Select 2,finefeed,>,2,target,ASC
```

8.13.1.1.3. Descending Order

Same as SQL command: SELECT * FROM A2 WHERE finefeed > 2 ORDER BY target DESC. The values are returned in Descending Order:

```
Select 2,finefeed,>,2,target,DESC
```

9 Miscellaneous Commands

9.1. Command Reference

- Note: For the Node property in common commands, the only legal value for the IND500x is 0.

Command	Usage
ADDTIME	Adds days, hours, minutes and years to a double floating point representation of date and time.
ArrayCopy	Copy the contents of one numeric array to another.
Get Task ID	Gets the ID of the running task.
JulDate	Returns a double floating point variable representation of date and time, converted from a string.
Language	Retrieves a message from the custom message database in the currently selected language.
Resume Task	Resumes selected TaskExpert application.
Security	Reads the iButton EEPROM security code to validate the security code to verify that this terminal has been authorized to run this application.
Start Task	Starts selected TaskExpert application.
Stop Task	Stops a selected TaskExpert application.
Suspend Task	Puts selected TaskExpert application in a wait state until a Resume Task command is issued.
TimDat	Returns a string representation of the date and time that it converts from the double floating-point representation.

9.2. ADDTIME

Miscellaneous ADDTIME

AddTime() adds days, hours, minutes, and years to a double floating-point representation of date and time.

9.2.1. Properties

OldTime	[Double] Double floating-point representation of the date and time.
Days	[Integer] Number of days to add to the old date/time.
Hours	[Integer, Optional] Number of hours to add to the old date/time.

Minutes	[Integer, Optional] Number of minutes to add to the old date/time.
Years	[Integer, Optional] Number of years to add to the old date/time.
NewTime	[Double variable, Output/Result variable] The return value is a double floating-point representation of the new date and time.

9.3. ArrayCopy

Miscellaneous ArrayCopy

Copy the contents of one numeric array to another.

9.3.1. Properties

Source Name	[String] Name of the source array to copy data from.
Destination Name	[String] Name of the destination array to copy data to.
Length	[Integer] Number of array elements to copy. Default is all.
Source Offset	[Integer] Optional index location into the source array to start the copy.
Destination Offset	[Integer] Optional index location into the destination array to start the copy.
Status	`[Integer] Number of elements copied.`

9.4. Get Task ID

Miscellaneous Get Task ID

Get Task ID of this running task.

9.4.1. Properties

Task ID	[Integer variable, Output/Result variable] Returns the task slot. Possible return values are 1 to 3.

9.5. JulDate

Miscellaneous JulDate

The JulDate() function returns a double floating-point variable representation of the date and time that it converts from a string representation.

9.5.1. Parameters

The calling parameter is a string in the format "yyyy-mm-dd hh:mm:ss"

- Note: There must be a space between the date and the time.

9.5.1.1. Return Value

The return value is a double floating-point representation of the date and time. The format of the return value is "YYYYDDD.<fractional seconds of the day>". The fractional seconds of the day are calculated as follows:

Total Number of Seconds in a Day:

24h*60m*60s = 86400s

Current Number of Seconds in the Day (using example below):

Current Time = 09:55:23
9h = 32400s
55m = 3300s
23s = 23s

Total Current Number of Seconds = 35723s

Fractional Seconds of the Day = (Current Seconds + 0.5)/Total Seconds

35723.5/86400 = 0.413466435

- Note: The 0.5 is added for rounding.

9.5.1.2. Example

```
Datetime$ = "2009-01-29 09:55:23"

MyDatetime# = JulDate(Datetime$)
```

9.5.1.3. Output

```
2009029.413466435
```

9.6. Language

Miscellaneous Language

Retrieve a message from the custom message database in the currently selected language.

9.6.1. Properties

Index	[Integer] Message index.
Text	[String variable, Output/Result variable] Returned message text.

9.7. Resume Task

Miscellaneous Resume Task

Resume selected TaskExpert application. For this function to have any effect, the selected application must be in a suspended state.

9.7.1. Properties

Task ID	[Integer] Selects the task slot. Legal values are 1 to 3.
Node	[Integer, Optional] Selects the node in the cluster. Legal values are 0. 0 refers to the local node.
Status	[Integer variable, Output/Result variable, Optional]

9.8. Security

Read the iButton EEPROM security code. The application can validate the security code to verify that this terminal has been authorized to run this application.

9.8.1. Properties

Security Array	[Integer variable, Output/Result variable] Contains the security string read from the EEPROM. It must be an integer array.

9.9. Start Task

Start selected TaskExpert application. Optionally, you can specify a TaskExpert program name.

9.9.1. Properties

TaskID	[Integer] Selects the task slot. Legal values are 1 to 3.
Node	[Integer] Selects the node in the cluster. Legal value is 0.
Program Name	[Integer] Optionally selects the TaskExpert application to run in this slot. If you specify a program name, the TaskExpert Interpreter will write this program name into the "aq" block of Shared Data and will attempt to start that program in the specified slot. If you do not specify a program name, the TaskExpert Interpreter will attempt to start the application already defined in the "aq" block of Shared Data.
Task Function Status	[Integer variable, Output/Result variable, Optional] Return value of this function block.

Return Status:

TASK FUNCTION STATUS	
TASK FUNCTION COMPLETED SUCCESSFULLY	0
INVALID TASK FUNCTION PARAMETER	98
CANNOT ACCESS TASK SD TRIGGER	99

9.10. Stop Task

Stop a selected TaskExpert application. Stop terminates the application.

9.10.1. Properties

Task ID	[Integer] Selects the task slot. Legal values are 1 to 3.
Node	[Integer] Selects the node in the cluster. Legal values are 0. 0 refers to the local node.
Status	[Integer variable, Output/Result variable, Optional] Return value of this function block.

9.11. Suspend Task

Suspend selected TaskExpert application. The Suspend Function puts the task in a wait state until you issue a command to resume it.

9.11.1. Properties

Task ID	[Integer] Selects the task slot. Legal values are 1 to 3.
Node	[Integer, Optional] Selects the node in the cluster. Legal values are 0. 0 refers to the local node.
Status	[Integer variable, Output/Result variable, Optional] Return value of this function block.

9.12. TimDat

The TIMDAT function returns a string representation of the date and time that it converts from the double floating-point representation.

9.12.1. Properties

Time	[Double] Floating-point variable to be converted to a date string.
Datetime	[String variable, Output/Result variable] The return value is a string in the following format: yyyy-mm-dd hh:mm:ss

10 Display Object Commands

The Display Objects tools are WYSIWYG versions of the Display and Keyboard functions. They may include complex combinations of those individual commands. These objects create the same compiled code as the Display and Keyboard functions, but also provide a special visual design object that allows the developer to see the results of these functions.

10.1. Command Reference

Display Commands

Command	Usage
Combobox	Displays Combobox on the screen in the application window.
Draw Image	Draws a graphic image in the application window.
Draw Shared Data Function	Displays contents of Shared Data in the application window.
ImageSD	Draws a variable graphic image in the application window
Popup Box	Draws a POPUP Box in the application window.
System Message	Writes critical error message to the System error line.
Textbox	Displays the text entry box on the screen in the application window.
Textbox and Label	Displays text entry box with a label on the screen in the application window.

10.2. Combobox

Display Objects Combobox

Displays the Combo Box on the screen in the application window. The application window appears immediately below the Weight/SmartTrac display and above the SoftKey display. After the operator moves the focus to the Combo Box on the screen using the navigation keys, he may select one of a number of selections from the Combo Box. The Combo Box is a list of selections that allows the operator to select one from the list. The operator makes his selection via the arrow keys and terminates the selection using the Enter Key. Focus moves to the next entry in the Focus List.

While focus is on the ComboBox, the arrow keys change the current selection. The up and left arrows both move the selection up, and the down and right arrows both move the selection down.

- Note: The TaskExpert variables associated with the Combobox are 40-character strings. The maximum number of characters returned by a Combobox will be 39, since one character is the null terminator.

10.2.1. Properties

Index	[Integer] The index of the display object in the application display. Legal values are 1 to 19. Object 20 is reserved for Single-Line Data Entry Line Prompt.
Index Variable Instance	[Integer constant only, Optional] Index of the shared variables to be associated. Legal values are 1-19.
X-Coordinate	[Integer] Position of the text display relative to the top, left-hand corner of the application display window.
Y-Coordinate	[Integer] Position of the text display relative to the top, left-hand corner of the application display window.
Width	[Integer] Width of the combobox item.
Selection List	[Optional] A comma-separated list of selections from which the operator may choose. Note: The Selection List string is limited to 160 characters, including the commas to separate the list options.
Font	[Optional] Font type and size of the display.
Color	[Optional] Select text color. Options: **Blue** (default), Cyan, Green, White, Black, Dark Gray, Normal Gray, Light Gray, Red, Orange, Yellow
Default	[String, Optional] The selection from Selection List that the COMBOBOX has initially selected when it is first displayed.

Status	[Integer variable, Output/Result variable, Optional]
	Return value of the this function block.
	Return Status:
	`SUCCESS 0`
	`ERR_TOO_MANY_DISPLAY_OBJECTS -1`
	`ERR_OBJECT_NOT_ALLOCATED -4`
	`ERR_DISPLAY_UNICODE_CONVERSION -7`
	`ERR_XY_OUT_OF_RANGE -8`

VariableName	[Optional] Associated shared variable name, this variable name is mapped to shared variable "tx01XX" where XX=Display Object Index.
AssociatedEvent	[Function name in the current project, Optional] Subroutine to call when data in Combobox is changed.

10.2.2. Returned Data

After the operator makes his selection using the navigation keys, TaskExpert returns the number of the selected entry and the selected data to the application, comma-separated, in a Shared Data field, tx0101 through tx0120. The specific Shared Data field corresponds to the object index% specified in the function call. The application can set an event on this Shared Data field so that TaskExpert alerts the application when the data entry is complete.

TaskExpert also returns the specific data upon creation of the ComboBox. If the application has registered an event, it will get the event trigger upon creation.

10.3. Draw Image

Draw a graphic image in the application window, defining its attributes.

The application window appears immediately below the Weight/SmartTrac display and above the SoftKey display. There can be up to 20 display objects in the application display, where an object is a text or graphical display. The (x,y) coordinates for each object are relative to the top, left corner of the application window. The application can overwrite an existing object by executing a display command with the new attributes for the object.

Please remember only one TaskExpert application at a time can interface to the operator console.

10.3.1. Properties

Index	[Integer] Index of the display object in the application display. Legal values are 1 to 19. Object 20 is reserved for Single-Line Data Entry Line Prompt.
X-Coordinate	[Integer] X-Coordinate is the position of the image display relative to the top, left-hand corner of the application display window.
Y-Coordinate	[Integer] Y-Coordinate is the position of the image display relative to the top, left-hand corner of the application display window.
Local Image	[Optional] Image file name to display. File must be available in Local Machine.
File Name	[String] Name of the file where the bitmap of the graphic display image resides.
Status	[Integer variable, Output/Result variable, Optional] Return value of the this function block.

Return Status:

SUCCESS	0
ERR_TOO_MANY_DISPLAY_OBJECTS	-1
ERR_CANNOT_ACCESS_FILE	-2
ERR_INVALID_HANDLE	-3
ERR_OBJECT_NOT_ALLOCATED	-4
ERR_INVALID_SHARED_DATA	-5
ERR_CANNOT_SCROLL	-6
ERR_DISPLAY_UNICODE_CONVERSION	-7
ERR_XY_OUT_OF_RANGE	-8
ERR_INVALID_TASK_EXPERT_INSTANCE	-9

10.4. Draw Shared Variable

Display the contents of Shared Data in the application window. The application window appears immediately below the Weight/SmartTrac display and above the SoftKey display. If the Shared Data field is a "callback" Shared Data field, the function automatically updates display whenever the contents of the display changes.

Please remember only one TaskExpert application at a time can interface to the operator console.

10.4.1. Properties

Index	[Integer] Index of the display object in the application display. Legal values are 1 to 19. Object 20 is reserved for Single-Line Data Entry Line Prompt.
X-Coordinate	[Integer] X-Coordinate is the position of the text display relative to the top, left-hand corner of the application display window.
Y-Coordinate	[Integer] Y-Coordinate is the position of the text display relative to the top, left-hand corner of the application display window.
SD Name	Six-character name of the Shared Data field.
Font	[Optional] Font type and size of the display.
Color	[Optional] Select text color. Options: **Blue** (default), Cyan, Green, White, Black, Dark Gray, Normal Gray, Light Gray, Red, Orange, Yellow
Format	Defines the format of the displayed data. For numeric data, "#nn.dd" is the format specification where "nn" is max number of numeric digits & "dd" is decimal point position. For alphanumeric string data, "!ss.t" is the format specification where "ss" is maximum number of alphanumeric characters to display. "t" defines how to trim blank characters in the string. 1= trim off leading blanks; 2 = trim off trailing blanks; 3 = trim off both leading and trailing blanks. The default is "!0.0" where 0 length implies display all characters in the string and 0 trim implies no trimming of the blank characters.
Status	[Integer variable, Output/Result variable, Optional] Return value of the this function block.

Return Status:

```
SUCCESS                                   0
SUCCESS - SD PREVIOUSLY USED              1
ERR_TOO_MANY_DISPLAY_OBJECTS             -1
ERR_CANNOT_ACCESS_FILE                   -2
ERR_INVALID_HANDLE                       -3
ERR_OBJECT_NOT_ALLOCATED                 -4
ERR_INVALID_SHARED_DATA                  -5
ERR_CANNOT_SCROLL                        -6
ERR_DISPLAY_UNICODE_CONVERSION           -7
ERR_XY_OUT_OF_RANGE                      -8
ERR_INVALID_TASK_EXPERT_INSTANCE         -9
```

10.5. ImageSD

Display & Keyboard ImageSD

Draw a variable, changing graphic image in the application window. Upon executing the IMAGESD command, TaskExpert selects the graphic image from a list of up to 6 bitmaps, based on the value in a Shared Data field. TaskExpert automatically switches the display to a new graphic image whenever the Shared Data value changes without further commands from the application program.

The application window appears in the display immediately below the Weight/SmartTrac display and above the SoftKey display. There can be up to 20 display objects in the application display, where an object is a text or graphical display. The (x,y) coordinates for each object are relative to the top, left corner of the application window. The application can overwrite an existing object by executing a display command with the new attributes for the object.

Please remember only one TaskExpert application at a time can interface to the operator console.

10.5.1. Properties

Index	[Integer] The index of the display object in the application display. Legal values are 1 to 19. Object 20 is reserved for the Data Entry Line.
X-Coordinate	[Integer] The position of the image display relative to the top left-hand corner of the application display window.
Y-Coordinate	[Integer] The position of the image display relative to the top left-hand corner of the application display window.
SD Name	[Integer] The six-character name of the Shared Data field, which must be of type Bl, By, US or UL, and be a "callback" field.
Error Filename	[String] The name of the bitmap file which should be displayed when the Shared Data field's value does not match one of the values for which a bitmap is specified in Filename0 - Filename4
Local Filename0	[Optional] File name of bitmap image to display when the Shared Data value is 0. File must be available in Local machine.
Filename0	[String] File name of bitmap image which should be displayed when the Shared Data field's value is 0.
Local Filename1	[Optional] File name of bitmap image to display when the Shared Data value is 1. File must be available in Local machine.
Filename1	[String] File name of bitmap image which should be displayed when the Shared Data field's value is 1.
Local Filename2	[Optional] File name of bitmap image to display when the Shared Data value is 2. File must be available in Local machine.
Filename2	[String] File name of bitmap image which should be displayed when the Shared Data field's value is 2.
Local Filename3	[Optional] File name of bitmap image to display when the Shared Data value is 3. File must be available in Local machine.
Filename3	[String] File name of bitmap image which should be displayed when the Shared Data field's value is 3.
Local Filename4	[Optional] File name of bitmap image to display when the Shared Data value is 4. File must be available in Local machine.
Filename4	[String] File name of bitmap image which should be displayed when the Shared Data field's value is 4.

Status	[Integer variable, Output/Result variable, Optional] Return value of this function block.

Return Status:

```
SUCCESS                                  0
ERR_TOO_MANY_DISPLAY_OBJECTS            -1
ERR_CANNOT_ACCESS_FILE                  -2
ERR_INVALID_HANDLE                      -3
ERR_OBJECT_NOT_ALLOCATED                -4
ERR_INVALID_SHARED_DATA                 -5
ERR_CANNOT_SCROLL                       -6
ERR_DISPLAY_UNICODE_CONVERSION          -7
ERR_XY_OUT_OF_RANGE                     -8
ERR_INVALID_TASK_EXPERT_INSTANCE        -9
```

10.6. Popup Box Function

Display Objects — Popup Display

Draws a POPUP Box in the application window. The operator must acknowledge the PopUp message by hitting the enter key before TaskExpert program execution continues. The PopUp has a title and two text strings. Figure 10-1 shows a collapsed Popup Display object, and a Popup function block.

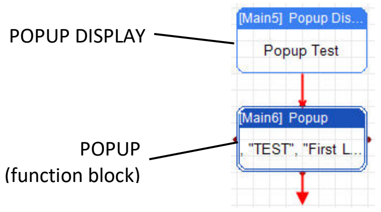


Figure 10-1: Collapsed Popup Display and Popup Function

The Popup Display can be expanded (Figure 10-2) by double clicking on the collapsed block.

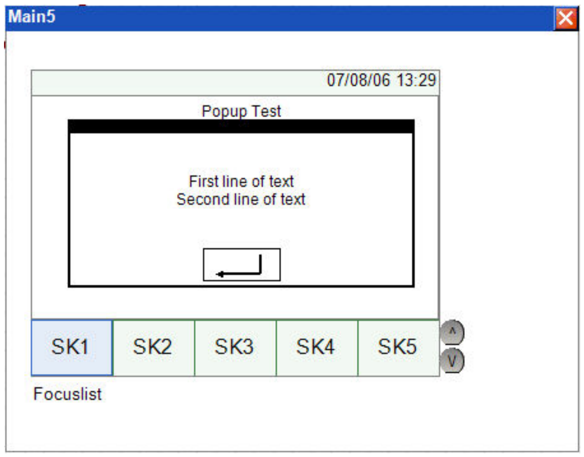


Figure 10-2: Expanded Popup Display

10.6.1. Properties

Title	[String] Title of the POPUP box.

Text1	[String] First text message in the POPUP box.
Text2	[String, Optional] Second text message in the POPUP box. Its default value is "Press ENTER to continue". A newline character ("\n") is used to insert a new line into the text. Text2 supports a maximum of 4 lines. The maximum number of character in each line is 31. The box size expand automatically according to the number of lines.
Status	[Integer variable, Output/Result variable, Optional] Return value of this function block.

10.7. System Message Display

Display Objects System Message Display

Write critical error message to the System error line. Applications must only use this command for display critical system failures, especially, failures that may cause safety hazards and failures that require immediate operator attention.

10.7.1. Properties

Text	[String] Text of the message to be written to the System error line.
Priority	[Integer] Priority of the error message.
Status	[Integer variable, Output/Result variable, Optional] Return value of this function block.

10.8. Text Box

Display Objects Text Box

Display the text entry box on the screen in the application window. The application window appears immediately below the Weight/SmartTrac display and above the SoftKey display. After the operator moves the focus to the text entry box on the screen using the navigation keys, he may enter data through the keypad or keyboard into the text box. He may terminate the entry with the Enter key.

While the focus is on the textbox, the arrow keys move the cursor within the textbox. The left arrow moves the cursor to the left, and the right arrow both moves the cursor to the right.

When the TEXTBOX receives the focus, it selects the entire contents so the first key pressed replaces the entire contents of the textbox.

- Note: The TaskExpert variables associated with the Textbox are 40-character strings. The maximum number of characters returned by a Textbox will be 39, since one character is the null terminator.

Please remember only one TaskExpert application can interface to the operator console.

10.8.1. Properties

Index	[Integer] Index of the display object in the application display. Legal values are 1 to 19. Object 20 is reserved for Single-Line Data Entry Line Prompt.
Index Variable Instance	[Integer, Optional] Index of the shared variable to be associated. Legal values are 1 to 19.
Variable name	[Optional] Associated shared variable name. This name is mapped to shared variable "tx01XX" where XX = the Display Object Index.
Associated Event	[Function name in the current project, Optional] Subroutine to call when data in Textbox is changed.
X-Coordinate	Position of the text display relative to the top, left-hand corner of the application display window.
Y-Coordinate	Position of the text display relative to the top, left-hand corner of the application display window.
Width	[Integer] Width of the TEXTBOX
Default	Default text displayed in the TEXTBOX
Format	[String, Optional] Defines the format of the entered-data. In numeric data entry mode, "#nn.dd" is the format specification where nn is max number of numeric digits & dd is decimal point position. The numeric data entered into the Text Box appears from right-to-left, filling in behind the decimal point first. In alphanumeric data entry mode, "!ss" is the format specification where ss is maximum number of alphanumeric characters. Data entered into the Text Box appears in left-to-right order. TaskExpert automatically enables alphanumeric data entry through the keypad when the focus comes onto the text box. In alphanumeric password entry mode, "*ss" specifies a format where ss is the maximum number of alphanumeric characters, and entered characters appear as asterisks (*). Data appears in left-to-right order. TaskExpert does not check for ranges and max number of decimal places until the operator presses the Enter key. If there is a problem with the entry the application shows a PopUp error and, after operator acknowledges the PopUp, returns focus to the control.
Font	[Optional] Font type and size of the display.
Color	[Optional] Select text color. Options: **Blue** (default), Cyan, Green, White, Black, Dark Gray, Normal Gray, Light Gray, Red, Orange, Yellow
Status	[Integer variable, Output/Result variable, Optional] Return value of this function block. Return Status:

```
SUCCESS                          0
ERR_TOO_MANY_DISPLAY_OBJECTS    -1
ERR_OBJECT_NOT_ALLOCATED        -4
ERR_DISPLAY_UNICODE_CONVERSION  -7
ERR_XY_OUT_OF_RANGE             -8
```

After the operator enters the data and presses either the enter key or one of the navigation keys, TaskExpert returns the data to the application in a Shared Data fields, tx0101 through tx0120. The specific Shared Data field corresponds to the object index% specified in the function call. The application can set an event on the Shared Data field so that TaskExpert alerts the application when the data entry is complete.

TaskExpert also returns the specific data upon creation of the TextBox. If the application has registered an event, it will get the event trigger upon creation.

10.9. Text Box and Label

Display the text entry box with a label on the screen in the application window. The application window appears immediately below the Weight/SmartTrac display and above the SoftKey display. After the operator moves the focus to the text entry box on the screen using the navigation keys, he may enter data through the keypad or keyboard into the text box. He may terminate the entry with the Enter key.

While the focus is on the textbox, the arrow keys move the cursor within the textbox. The left arrow moves the cursor to the left, and the right arrow both moves the cursor to the right if the format is alphanumeric (i.e. "!ss") or left justified numeric (i.e. "%nn"). If the format dictates right justified numeric entry (i.e. "#nn.dd") the arrow keys have no effect.

When the TEXTBOX receives the focus, it selects the entire contents so the first key pressed replaces the entire contents of the textbox.

- Note: The TaskExpert variables associated with the Textbox are 40-character strings. The maximum number of characters returned by a Textbox will be 39, since one character is the null terminator.

Please remember only one TaskExpert application can interface to the operator console.

10.9.1. Properties

Index	[Integer] Index of the display object in the application display. Legal values are 1 to 19. Object 20 is reserved for Single-Line Data Entry Line Prompt.
Index Variable Instance	[Integer, Optional] Index of the shared variable to be associated. Legal values are 1 to 19.
Variable name	[Optional] Associated shared variable name. This name is mapped to shared variable "tx01XX" where XX = the Display Object Index.
Associated Event	[Function name in the current project, Optional] Subroutine to call when data in Textbox is changed.
X-Coordinate	Position of the text display relative to the top, left-hand corner of the application display window.
Y-Coordinate	Position of the text display relative to the top, left-hand corner of the application display window.
Width	[Integer] Width of the TEXTBOX
Default	Default text displayed in the TEXTBOX

Format	[String, Optional] Defines the format of the entered-data. In numeric data entry mode, "#nn.dd" is the format specification where nn is max number of numeric digits & dd is decimal point position. The numeric data entered into the Text Box appears from right-to-left, filling in behind the decimal point first. In alphanumeric data entry mode, "!ss" is the format specification where ss is maximum number of alphanumeric characters. Data entered into the Text Box appears in left-to-right order. TaskExpert automatically enables alphanumeric data entry through the keypad when the focus comes onto the text box. In alphanumeric password entry mode, "*ss" specifies a format where ss is the maximum number of alphanumeric characters, and entered characters appear as asterisks (*). Data appears in left-to-right order.
Font	[Optional] Font type and size of the display.
Color	[Optional] Select text color. Options: Black (default

Status	[Integer variable, Output/Result variable, Optional] Return value of this function block.
	Return Status:
	`SUCCESS                          0`
	`ERR_TOO_MANY_DISPLAY_OBJECTS    -1`
	`ERR_OBJECT_NOT_ALLOCATED        -4`
	`ERR_DISPLAY_UNICODE_CONVERSION  -7`
	`ERR_XY_OUT_OF_RANGE             -8`

Label Index	[Integer] Index of the display object in the application display. Legal values are 1 to 19.
Label X-Coordinate	[Integer] X-Coordinate is position of the text display relative to the top, left-hand corner of the application display window.
Label Y-Coordinate	[Integer] Y-Coordinate is position of the text display relative to the top, left-hand corner of the application display window.
Label Text	[Optional] Text to display.
Label Text ID	[Integer, Optional] Text ID used to fetch text from language database available in the Terminal.
Label Font	[Optional] Select the desired Font from list.
Label Color	[Optional] Select text color from list.
Label Status	[Integer variable, Output/Result variable, Optional] Return value of this function block.

After the operator enters the data and presses either the enter key or one of the navigation keys, TaskExpert returns the data to the application in a Shared Data fields, tx0101 through tx0120. The specific Shared Data field corresponds to the object index% specified in the function call. The application can set an event on the Shared Data field so that TaskExpert alerts the application when the data entry is complete.

TaskExpert also returns the specific data upon creation of the TextBox. If the application has registered an event, it will get the event trigger upon creation.

11 Event and Error Handling

This section discusses error messages that may be output to the LPRINT device during debugging or program execution. The terminal display will show the Error Number Code and Line Number, with the error message being output to the LPRINT device (TaskExpert PC tool, a printer, a PC running a communication emulation program). For example, the error Device Error would show up on the terminal's system line display. The message output to the LPRINT device should show as:

```
TaskExpert Program Error:  Line 15:  Device Error.
```

11.1. TaskExpert Error Codes

The following table lists possible error codes and messages in TaskExpert.

Error Code	Error Message	Description	Probable Cause	Remedy	Trapped with ON ERROR*
0	Failed to open file	TaskExpert programming error.	TaskExpert attempted to open a nonexistent file or serial communications device.	Correct TaskExpert program.	•
1	Failed to find memory	TaskExpert programming error.	TaskExpert exceeded the memory limits of the system.	Reduce lines. Eliminate unnecessary spaces in program. Reduce variables. Reduce size of arrays. When chaining TaskExpert programs, chain in the largest program first to reduce memory fragmentation.	
2	Invalid Line Number	TaskExpert programming error.	TaskExpert contains a line number greater than 30000 or is a duplicate of an existing line number.	Correct TaskExpert program.	
3	Resource in use	TaskExpert programming error.	TaskExpert tried to access a system resource in use by another terminal task. TaskExpert cannot open a serial port that has been assigned to a serial port connection in setup. When two or more TaskExpert applications share a serial port, only one can have the port open at a time.	Correct TaskExpert application. To share serial ports between multiple TaskExpert applications, develop sharing logic that checks for this specific error code.	•
4	LOAD: No File Specified	Operator error.	The LOAD command does not contain a file name.	Correct the command.	
5	No Line Number	TaskExpert programming error.	The program line does not have a line number.	Correct TaskExpert program.	

Error Code	Error Message	Description	Probable Cause	Remedy	Trapped with ON ERROR*
6	Indexed Record Not Found	TaskExpert programming error.	A record specified in a GET statement for an indexed sequential file could not be found in the file.	There should be an ON ERROR statement in the TaskExpert program to handle these potential situations.	●
7	RETURN without GOSUB	TaskExpert programming error.	RETURN statement is present without required GOSUB.	Correct TaskExpert program.	
8	Incomplete statement	TaskExpert programming error.	TaskExpert program contains a line that does not have the full syntax required for a line.	Correct TaskExpert program.	
9	ON without GOTO or GOSUB	TaskExpert programming error.	ON statement is present without required GOSUB.	Correct TaskExpert program.	
10	Value Is Out Range	TaskExpert programming error.	The TaskExpert statement is referring to a value out of the range of acceptable values.	Correct TaskExpert program.	
11	Syntax Error	TaskExpert programming error.	The TaskExpert program has a syntax error.	Correct TaskExpert program.	
12	Invalid Device Number	TaskExpert programming error.	The TaskExpert program is referencing a device # that is not open.	Correct TaskExpert program.	
13	Device error	TaskExpert programming error.	The TaskExpert program has referred to an illegal device or a device that is not open.	Correct TaskExpert program.	●
14	Error in Operating System Command	An error occurred in trying to access a file off of the Storage Card.	You tried to access a file that does not exist or the file system has been corrupted.	Verify the Flash2:\ memory. If the file system has been corrupted, re-initialize the files and rebuild them from the backup files you are maintaining on a PC.	●
15	Argument Must Be A String	TaskExpert programming error.	The argument being passed to the command was not of the variable type string.	Correct TaskExpert program.	
16	Event Definition Error	TaskExpert programming error.	Programming error in defining an event.	Correct TaskExpert program.	
17	Type Mismatch	TaskExpert programming error.	TaskExpert statement is using an invalid data type or is relating two incompatible data types.	Correct TaskExpert program.	●
18	Argument Is Not An Array Name	TaskExpert programming error.	TaskExpert program has attempted to dimension a variable that is not an array.	Correct TaskExpert program.	
19	Out Of Data	TaskExpert programming error.	TaskExpert program has issued more READ commands to initialize system variables than data specified in DATA statements.	Correct TaskExpert program.	

Error Code	Error Message	Description	Probable Cause	Remedy	Trapped with ON ERROR*
20	Overflow	TaskExpert programming error.	A TaskExpert program causes an overflow error by exceeding certain system limits. If you try to nest subroutines or loops beyond the terminal's limit, you get an overflow error. (Refer to Appendix B for the limits). Overflow errors can also be caused by syntax errors.	Correct TaskExpert program.	
21	Improper FOR-NEXT	TaskExpert programming error.	There is a NEXT statement without the required FOR statement.	Correct TaskExpert program.	
22	Undefined Function	TaskExpert programming error.	The TaskExpert statement is referring to an undefined function.	Correct TaskExpert program.	
23	Divide By Zero	TaskExpert programming error.	TaskExpert program attempted to divide a number by zero.	Correct TaskExpert program.	
24	Variable Cannot Be Redimensioned	TaskExpert programming error.	Once a TaskExpert application has declared a variable or an array, it cannot later be redimensioned to a different size array.	Correct TaskExpert program.	
25	OPTION BASE Must Be Called Prior to DIM	TaskExpert programming error.	The TaskExpert program must define the OPTION BASE before dimensioning an array.	Correct TaskExpert program.	
26	Illegal Command	TaskExpert programming error.	The TaskExpert program has issued a command that is not a legal command.	Correct TaskExpert program. Change the keyboard setting.	
27	Variable Has Too Many Dimensions	TaskExpert programming error.	TaskExpert arrays can have at most three dimensions.	Correct TaskExpert program.	
28	Invalid Shared Data Name	TaskExpert programming error.	The TaskExpert program is referencing an invalid Shared Data name.	Correct TaskExpert program.	•
29	Program Too Big	TaskExpert programming error.	The program exceeds the maximum number of text lines or file size.	For the first problem, separate the program into smaller files that can be run independently or be chained together. When chaining, always start execution with the largest program to avoid memory fragmentation.	
30	Line Too Big	TaskExpert programming error.	A TaskExpert line is greater than 1000 characters.	Correct TaskExpert program.	
31	Shared Data String Too Long	TaskExpert programming error.	TaskExpert can only access shared data fields whose length is less than the maximum TaskExpert string size.	Correct TaskExpert program.	•

Error Code	Error Message	Description	Probable Cause	Remedy	Trapped with ON ERROR*
32	No Remote Access	TaskExpert programming error.	The program is attempting to access a device that is already in use by a serial connection or by another TaskExpert program.	To access a serial device, you must remove all continuous output or input connections to the serial device in setup. To share a device among TaskExpert programs, you must setup a scheme where only one program has the device open at a time.	●
33	Unicode Conversion Error	Error in trying to access or write to a UNICODE file.	The file trying to be accessed does not contain UNICODE identifiers or the Input statement has an error occur when trying to convert byte string to UNICODE.	Verify file trying to be accessed is in UNICODE.	●
34	File Access Error	TaskExpert programming error.	This error occurs when trying to access a file to read or write and the command fails.	Correct TaskExpert program.	●
35	Database Already in Use	TaskExpert programming error.	This error occurs when the database has already been opened and the program attempts to open again.	Correct TaskExpert program.	●
36	Database Access Error	TaskExpert programming error.	Lower level access (system access) to the database. Error occurs with incorrect syntax to the database, such as trying to access a record that doesn't exist or issuing a query from the DataGrid object with incorrect syntax.	Correct TaskExpert program.	●
37	Invalid Database Index	TaskExpert programming error.	The database instance trying to be accessed has not yet been opened. Any command issued to access that instance will return this error.	Correct TaskExpert program.	●
38	SQL Command Error	TaskExpert programming error.	Error in SQL syntax.	Correct TaskExpert program.	●
39	TaskExpert Not Authorized	Hardware error.	Incorrect or no iButton installed in the terminal.	Install iButton or correct iButton into the terminal.	
40	Custom Application Not Authorized	Hardware error.	Incorrect iButton installed in the terminal.	Install the correct iButton into the terminal.	
41	Pac Application Not Authorized	Hardware error.	Incorrect iButton installed in the terminal.	Install the correct iButton into the terminal.	
42	Baseboard Switch 2 Set	SW2-1 is set ON.	Switch disables TaskExpert from running on Terminal.	Set switch into the OFF position.	
43	Application Not Authorized	Hardware error.	Incorrect iButton installed in the terminal.	Install the correct iButton into the terminal.	

* Errors indicated in this column can be trapped with the ON ERROR command, and are "recoverable" while continuing to run the application.

11.2. Clear Event

Event & Error Handling Clear Event

Clears outstanding event triggers. The TaskExpert interpreter automatically clears an event trigger upon completion of an event trapping routing for that trigger.

11.2.1. Properties

Variable	[Event variable, Optional] name of the specific event that you want to clear. If no event name is specified, all event triggers are cleared.

11.3. Defshr Event

Event & Error Handling Defshr Event

Allows the application to assign a shared data variable as an event while the application is running. This would typically be used if an event is deleted during application operation – it could be reassigned using this command.

11.4. Delete Event

Event & Error Handling Delete Event

De-allocates an event.

11.4.1. Properties

Variable	[Event variable, Optional] name of the specific event that you want to clear. If no event name is specified, all event triggers are deleted.

11.5. Disable

Event & Error Handling Disable

Disables asynchronous event triggers. This command is used to protect critical sections of code.

11.6. Enable

Event & Error Handling Enable

Re-enables asynchronous event triggers after a critical section of code.

11.7. Error

Event & Error Handling Error

Simulates an occurrence of an error. Used to debug error handling routines.

11.7.1. Properties

Error Code	[Integer] Simulates an occurrence of an error. Error code to throw.

11.8. Error Code

Event & Error Handling Error Code

Returns the runtime error code for the most recent error. Used in error handling routines to help identify the program and determine whether the program can recover from the error.

11.8.1. Properties

Error Code	[Integer variable, Output/Result variable] Returns the runtime error code for the most recent error.

11.9. Error Line

Event & Error Handling Error Line

Returns the line number where the error occurred or the closest line number before the line where the error occurred. Used as a debugging aid to fix runtime errors in you program.

11.9.1. Properties

Error Code	[Integer variable, Output/Result variable] Returns the line number where the error occurred or the closest line number before the line where the error occurred.

11.10. Event

Event & Error Handling Event

Allocates a keyboard event or timer event. An event occurs asynchronously from the normal execution of normal operation.

The keyboard event triggers an event when there is a key available. Use the INKEY function to read the key.

The timer event triggers at the expiration of the timer. Use the STARTIME command to start the timer.

11.10.1. Properties

Event	Select keyboard or timer event.

11.11. On Error

Enables error handling and when an error occurs directs the program to an error handling routing. If ON ERROR is not used, any runtime error ends the program.

11.11.1. Properties

Call Type	Type of ON ERROR call. Options: GOTO, Subroutine
Block/Function	Select the error handling block or error handling subroutine.

11.12. On Event

Enables you to asynchronously monitor an event and define the Event Service Routine. Upon the occurrence of an asynchronous event, the program execution branches to an event trapping subroutine.

Event trapping routines must be short routines that execute quickly and then return execution control back to the main program. The execution of an event trapping subroutine completes without interruption by another asynchronous event. The event trapping routines can occur between any two lines in the main program. Be careful of the variables used in these routines. Temporary variables, such as loop counters, should be unique to the event-trapping routine. Upon exit of the event-trapping routine, the TaskExpert interpreter automatically clears the event that triggered the execution of the routine.

11.12.1. Properties

Variable	[Event variable] Select the event variable for which a callback routine has to be registered.
Callback	[Function name in the current project] Subroutine to call when data in event variable is changed.

11.13. WaitEvent

Suspends program execution until an event trigger causes program execution to resume.

12 Ladder Commands

12.1. Command Reference

Command	Usage
New Ladder	Clears the current Ladder Program so you can start building a new Ladder Program.
Notes	Allows the user to add comments throughout an application.
RungAnd	Takes two inputs, "AND"s them together, and outputs the result.
RungAndNt	Takes two inputs, "AND"s them together, and outputs the inverse value.
RunMov	Takes an input and generates an output with the same value.
RungMovNt	Moves the inverse of the input to the output.
RungOr	Takes two inputs, "OR"s them together, and outputs the result.
RungOrNt	Takes two inputs, "OR"s them together, and outputs the inverse value.

12.2. Introduction

Ladder function blocks (rung statements) can be used to create a separate subroutine (Ladder routine) for ladder logic corresponding to the task. However rung statements can also be included anywhere in the flow of the program like normal blocks.

To add a ladder routine, right click on the active tab of the task and select the "Add Ladder routine" option. Rung Blocks can then be inserted between the ladder start and end blocks as in any other TaskExpert sequence.

There are three ways to add a new rung to the ladder:

- Drag and drop the rung statement between any two rung blocks.
- Select a rung block and double click on the toolbox item.
- Set focus on the rung block, then press the keyboard shortcut assigned to the rung statement.

Rung statements can also be copied and pasted across ordinary and ladder routines. Ladder routines can be included in libraries, and can use runtime instance variables.

Note: A project can have a maximum of 98 rung statements declared in it. This includes the rung statements available in the ladder routine(s), other routine(s) and those available in libraries.

The ladder start element has a "Display type" property (visible in the Properties panel at the right of the TaskExpert display). By default the display type is shared data name, as shown in Figure 12-1.

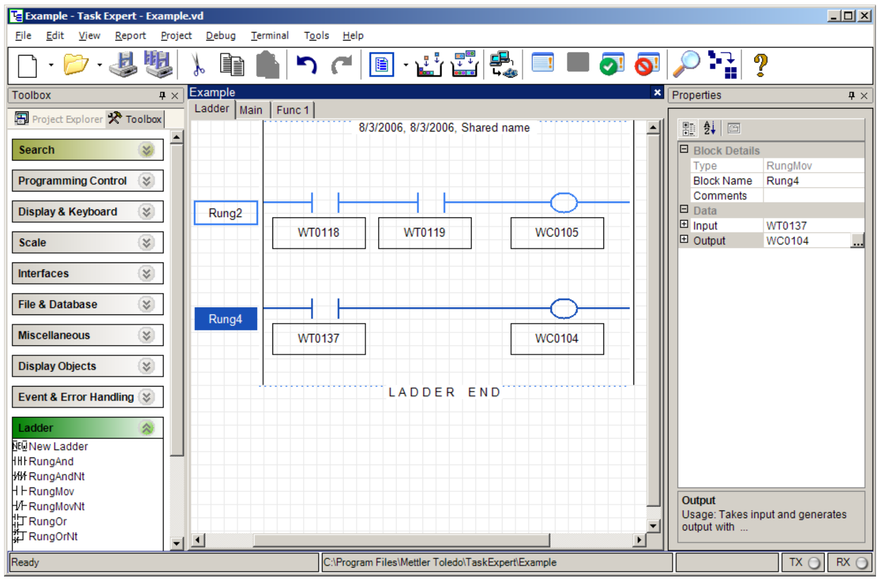


Figure 12-1: Ladder Routine, Shared Name Display Type

Setting the "Display Type" property to "Alias name" would display all the shared variables in the ladder in alias name as shown in Figure 12-2.

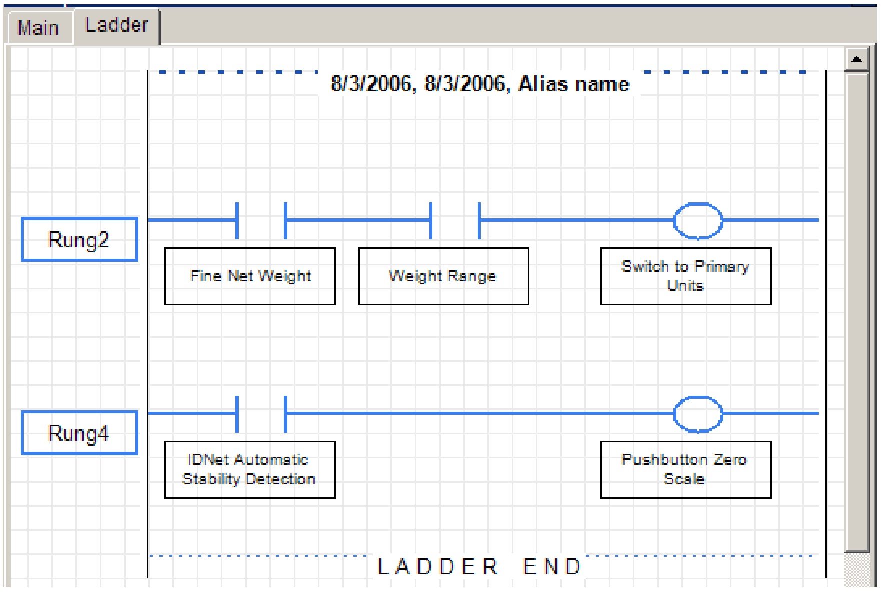


Figure 12-2: Ladder Routine, Alias Name Display Type

12.3. Ladder Logic

There are six Ladder Logic Rung commands. Each rung takes one or two inputs, and has one output. The rung inputs and outputs are physical Discrete IO or Shared Data commands.

The Ladder Logic Interpreter has an accumulator register, ACC, which you can use to implement more complex ladder programs. For example, you may need to AND three input signals together to generate one output signal. You can AND the first two signals together and put the result in the accumulator. Then, you can AND the accumulator with the third input signal to generate the output signal.

```
RUNGAND input1,input2,ACC
RUNGAND input3,ACC,output
```

When you build a Ladder Program with the Task Expert Interpreter, it places each successive Rung command at the end of the current Ladder Program in Shared Data.

12.3.1. New Ladder

Clears the current Ladder Program so you can start building a new Ladder Program.

- Note: It is not recommended to call the NewLadder command multiple times throughout a process. The command resets the ladder engine, which is located in flash memory. The common use for this command is to be called when starting up the program for the first time, to clear the ladder and set to new values using TaskExpert.

12.3.2. Notes

The NOTES block allows the user to add comments throughout an application. The block can be shown or hidden using the "show notes" option available in the Tools > TaskExpert Options > Appearance tab. This block can be placed anywhere within the development frame. NOTES are not executable code and are not included in the compiled code.

12.3.3. RungAnd

Takes two inputs, "AND's" them together, and outputs the result. For example, take a remote physical discrete input "permissive" signal and "AND" it with "Setpoint feeding" to generate a physical discrete output.

12.3.4. RungAndNt

Takes two inputs, "AND's" them together, and outputs the inverse value. For example, take two physical inputs and generate a physical discrete output.

12.3.5. RungMov

Takes an input and generates an output with the same value. For example, take a tare on the scale when a physical discrete input goes on.

12.3.6. RungMovNt

Moves the inverse of the input to the output. For example, turn on a physical discrete output when the data from scale is invalid.

12.3.7. RungOr

Takes two inputs, OR’s them together, and outputs the result. For example, turn on a physical discrete output if scale is in motion or under zero.

12.3.8. RungOrNt

Takes two inputs, OR’s them together, and outputs the inverse value. For example, turn on a physical discrete output when either the custom application turns off an application status or a physical discrete input is off.

12.4. Ladder Logic Interpreter

The IND500x has an internal Ladder Logic Interpreter that runs in the background to perform the continuously repetitive task of monitoring Discrete I/O. The Ladder Logic executes this task to minimize CPU utilization and to respond quickly in “real-time” to the changes in Discrete I/O or its associated Shared Data.

The Ladder Logic Interpreter runs in conjunction with the Task Expert Language. The Task Expert applications handle the sophisticated application tasks and operator interfaces. The Ladder Logic Interpreter handles efficiently the repetitive task of monitoring Discrete IO. It eliminates the significant processing overhead that would be required for Task Expert applications to accomplish this same task. The Task Expert applications and the Ladder Logic programs communicate to each other through Shared Data.

The Control Panel Setup application or Task Expert application program must build the Ladder Logic program required for their application. The Ladder Logic language has the simple commands described in the following section. The language provides flexibility for different applications to select what signals the Ladder Logic monitors and how to act on the signals. The Ladder Logic Program resides in Shared Data. It can have up to 98 rung commands.

The Ladder Logic Program only operates with local Shared Data and local Discrete I/O; it cannot access either Shared Data or Discrete IO remotely.

The Ladder Logic Interpreter has these operational features:

- The Ladder Logic Interpreter responds quickly in “real-time” to changes in Discrete I/O or Shared Data commands and statuses.
- It continuously monitors Discrete Inputs. When there is a state transition for a Discrete Input, it triggers the Shared Data command associated with the Discrete Input, according to the Ladder Logic program.

- It continuously monitors Discrete Output settings. When a Shared Data status associated with a Discrete Output changes polarity, it automatically changes the Discrete Output, according to the Ladder Logic program.
- The Control Panel Setup program automatically generates a simple Ladder Logic program that reflects the Discrete IO choices that the operator makes. For example, the operator may want to assign Discrete Input 1 to tare the scale.
- A Task Expert application program must build simple Ladder Logic programs in order to customize the monitoring of the Discrete I/O.

A Building a Custom Library

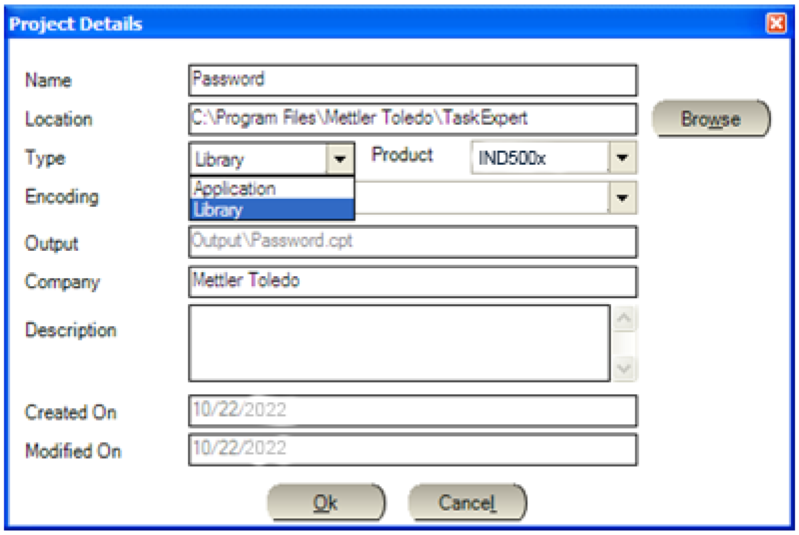
A.1. Overview

It is common in software development to have a collection of subroutines/methods and classes called Libraries. This collection can be accessed from any program that calls the library, passing in parameters and getting return parameters back from the subroutine or method. The same functionality is available in TaskExpert, by creating a common set of library functions and importing them into TaskExpert projects.

A.2. TaskExpert Capabilities

A.2.1. Creating a Library

Creating a library in TaskExpert is similar to creating an application, but the 'Type' parameter must be set to Library when creating a new project.



Name	Password
Location	C:\Program Files\Mettler Toledo\TaskExpert
Type	Library
Product	IND500x
Encoding	
Output	Output\Password.cpt
Company	Mettler Toledo
Description	
Created On	10/22/2022
Modified On	10/22/2022

Figure A-1: Selecting Library as Project Type

Once the project is open, the list of Entry Points is displayed in the Project Explorer window. Each Library project can contain multiple Entry Points subroutines, which are visible when the library is imported into a project.

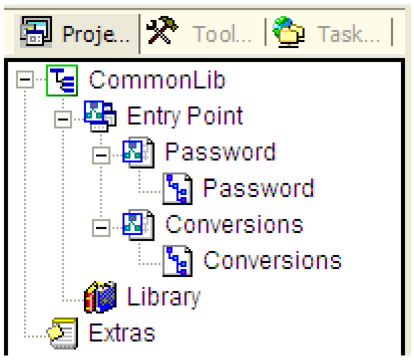
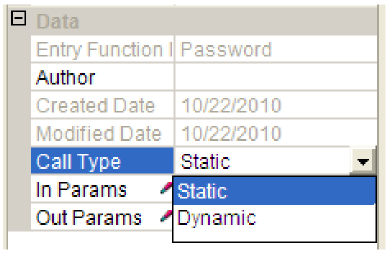


Figure A-2: Entry Points Displayed in Project Explorer

A.2.2. Library Types

Two types of library are available – Static and Dynamic. Use the Call Type parameter of the Library Start block to select the library type.



Data	
Entry Function I	Password
Author	
Created Date	10/22/2010
Modified Date	10/22/2010
Call Type	Static
In Params	Static
Out Params	Dynamic

Figure A-3: Selecting Type of Library

A.2.2.1. Static

A Static library automatically includes the input and output parameter variable in the TaskGlobal list when the library function is imported into a project. When the library function is called, the current values of the input variables are passed into the function and the output values are returned in the output variables. This type of library is typically used for direct TaskGlobal use.

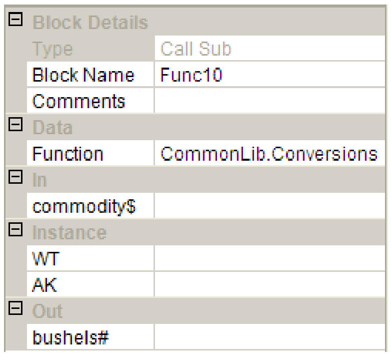
A.2.2.2. Dynamic

A Dynamic library requires values to be copied into temporary variables that are passed into the library function. This is common for copying shared data values directly to the function, such as the dynamic weight shared data block (wt--). For example, the live weight can be passed into the function, a calculation performed, and the result returned through the output variable. Again, though, the value must be copied from the output variable into another variable, either shared data or a TaskGlobal.

A.2.3. Runtime Shared Data Instance

Like application projects, shared data variables can be selected from the SD library and used within functions of the library. The SD library within a Library project contains a special instance selection called Runtime for certain variables. Selecting this special option will create another parameter for the library call that allows the instance value (integer) to be passed into the library at the time of the function call from the application.

The example function call shown in Figure A-4 displays both the WT and AK shared data blocks, where the instance integer can be specified.



Block Details	
Type	Call Sub
Block Name	Func10
Comments	
Data	
Function	CommonLib.Conversions
In	
commodity$	
Instance	
WT	
AK	
Out	
bushels#	

Figure A-4: Shared Data Instances

A.2.4. Importing a Library

To importa a Library into an Application project, right-click on the Library branch in the Project Explorer and select Import Library from the context menu.

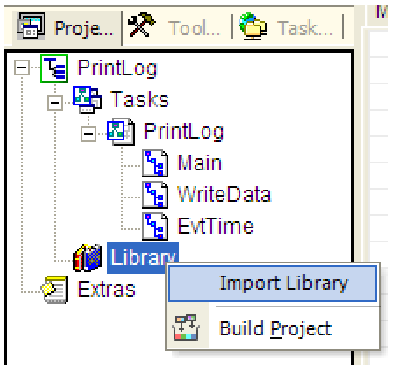
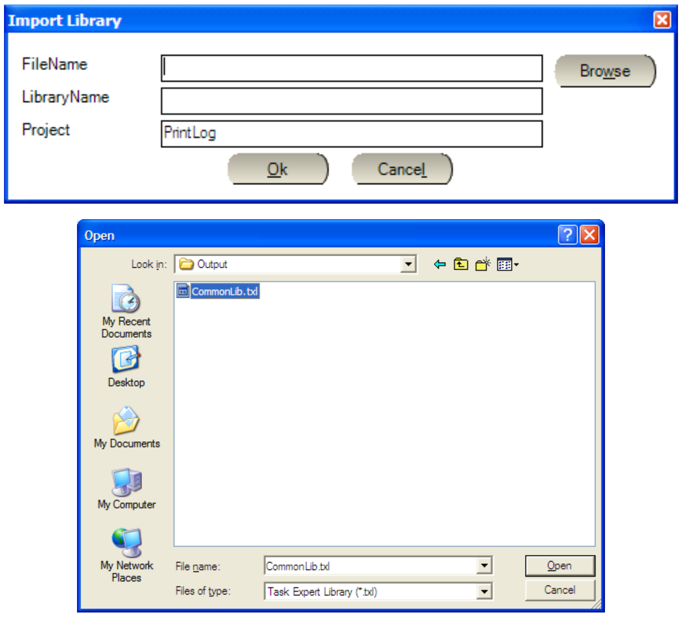


Figure A-5: Importing a Library

A series of dialog boxes appear, allowing the library to be selected from a location on the development machine. Once the library has been located, click OK to import it into the application project.



FileName
LibraryName
Project: PrintLog

Look in: Output
CommonLib.txt
File name: CommonLib.txt
Files of type: Task Expert Library (*.txt)

Figure A-6: Finding a Library to Import

Once the Library has been imported, the Library name, along with any associated functions, will display under the Library branch in the Project Explorer window.

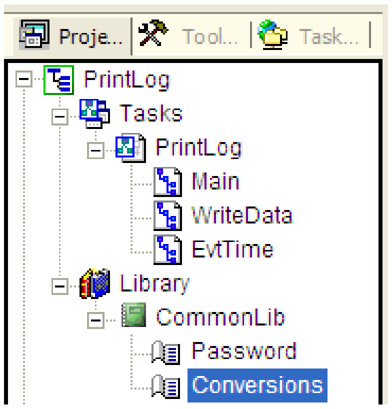
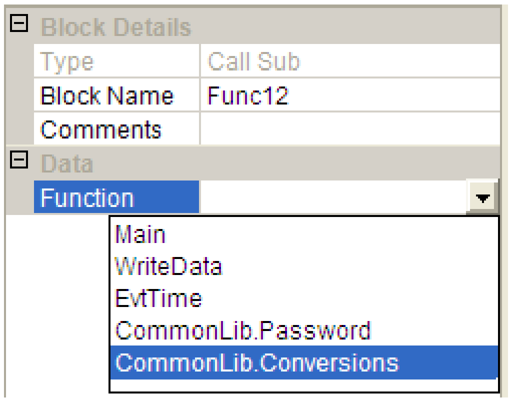


Figure A-7: Library and Associated Functions Displayed in Project Explorer

Calling the Library function is similar to calling a subroutine. Once the Library has been imported, the Library and its functions appear in the drop-down list for the CallSub function block.



Block Details	
Type	Call Sub
Block Name	Func12
Comments	
Data	
Function	

Figure A-8: Library Functions in Function Block Drop-Down Menu

B Maximum and Minimum Values for TaskExpert

The following table indicates the maximum and minimum values for TaskExpert items in the IND500x.

TaskExpert Item	Value
Maximum Number of Lines	7500
Maximum Program Size (bytes)	200k
Default Program Size (bytes)	1024
Maximum String Length (characters)	1001
Maximum Length for OPEN COM (characters)	1000
Maximum PLC Length (bytes)	500
Maximum Length for TaskGlobal or Global Variable Name	16
Maximum Dimensions for Array	3
Maximum Number of Variables	1500
Maximum Program Name Length	50
Maximum Filename Length	80
Maximum Appended Name Length	13
Number of Standard Tables	10
Maximum Number of Softkeys	15
Maximum Number of SD Instances	15
Maximum Number of GOSUB Levels	20
Maximum Number of WHILE Levels	20
Maximum Number of FOR Levels	20
Maximum Number of Elements in Expression Stack	64
Maximum Number of Debug Break Lines	50
Maximum Number of TCP/IP Sockets	12
Maximum Number of Input Keys	100
Maximum Number of TE Display Objects	20
Maximum Length of Display Message	160

C Softkeys

C.1. IND500x Softkeys

- Sizes are listed in **width x height** format.

C.1.1. Legends

Function	Size	Graphic
Motion	26x20	~
Centor of Zero	26x20	>0<
Range 1	33x 20	>|1|<
Range 2	33 x 20	>|2|<
Range 3	33 x 20	>|3|<
Unit	Noto sans Font H22, H26, H30, H36, H42 depending on display weight size	
Net / Gross indication	Noto sans Font H26	NET G
x10 Display	21 x 20	.05

C.1.2. Legends Extension

Function	Size	Graphic
Minweigh icon on	14x20	< <
Minweigh icon off	14x20	Flicker
Customized unit flag	20x20	*

C.1.3. Data Entry Display

Function	Size	Graphic
Numeric data entry	32x32	123
Upper case alpha entry	32x32	ABC

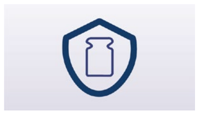
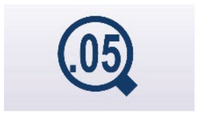
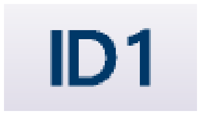
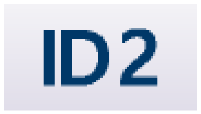
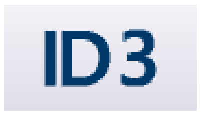
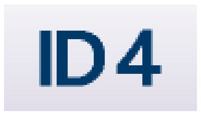
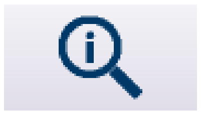
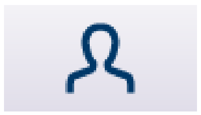
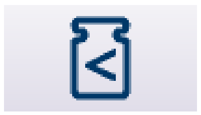
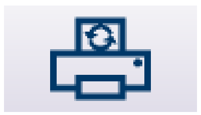
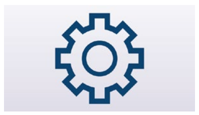
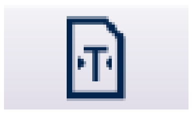
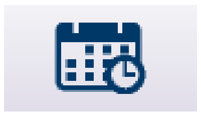
Function	Size	Graphic
Lower case alpha entry	32x32	✎ abc

C.1.4. Event Alert

Function	Size	Graphic
Event alert – service scheduled	88x49	🛡
Event alert – service due	88x49	🛡⚠
Event alert – perform service immediately	88x49	🛡✖

C.1.5. Home Screen Softkeys

Function	Size	Graphic
Blank	88x49	
Alibi Memory	88x49	A
Backlight Adjustment	88x49	☀
Comparators	88x49	→|←
Custom Print Trigger 1	88x49	1➔
Custom Print Trigger 2	88x49	2➔
Custom Pring Trigger 3	88x49	3➔

Function	Size	Graphic
Event Alert Recall	88x49	
Expand X10	88x49	
ID 1	88x49	
ID 2	88x49	
ID 3	88x49	
ID 4	88x49	
Information Recall	88x49	
Login	88x49	
MinWeigh	88x49	
Repeat Print	88x49	
Setup	88x49	
Tare Memory (Tare Table)	88x49	
Time & Date	88x49	

Function	Size	Graphic
Totals Recall	88x49	Σ
Transaction Counter	88x49	1 2 3
Unit Switching	88x49	

C.1.6. Information Recall

- Items marked with an asterisk ***** appear on a setup screen.

Function	Size	Graphic
Connected Devices	88x49	
Metrology Recall*****	88x49	M
Service	88x49	
System Info Recall*****	88x49	ⓘ
TaskExpert Checksum Recall*****	88x49	★
Terminal Status	88x49	i
Test Weight Information	88x49	i
Totals Recall	88x49	Σ
Transfer*****	88x49	⇪

Function	Size	Graphic
Weight Recall*****	88x49	⛉

C.1.7. Softkey Menu and Setup

Function	Size	Graphic
Enter Setup Menu	88x49	⚙
More Softkey Selections (Scroll left and right)	18x50	‹ ›

C.1.8. Calibration

- Items marked with an asterisk ***** appear on a setup screen.

Function	Size	Graphic
CalFREE Calibration*****	88x49	⌖
Zero Calibration*****	88x49	▸0◂
Span Calibration*****	88x49	▸|∨|◂
Step Calibration*****	88x49	▸▟◂
IDNet Service Mode*****	88x49	🔧
Start Calibration Test	88x49	▶
Test Weight Info*****	88x49	ℹ

Function	Size	Graphic
Skip* (Skips failed Calibration Test step and continues with the test)	88x49	⏭

C.1.9. Memory and Tables

- Items marked with an asterisk * appear on a setup screen.

Function	Size	Graphic
Clear	88x49	C
Clear Subtotal	88x49	C◇Σ
Clear Table Total	88x49	CΣ
Delete	88x49	🗑
Edit	88x49	✎
Insert New	88x49	🗋
Reset	88x49	↻
Search*	88x49	🔍
Tare* (Captures live scale weight and adds it to the Tare Table Record)	88x49	▸T◂
Transfer*	88x49	⇪

Function	Size	Graphic
Update	88x49	⟳
View Table*	88x49	🔍

C.1.10. Editing

- Items marked with an asterisk * appear on a setup screen.

Function	Size	Graphic
Clear*	88x49	C
Copy*	88x49	⧉
Delete*	88x49	🗑
Edit*	88x49	✎
Exit*	88x49	←
Insert New*	88x49	🗋
OK – accept change*	88x49	✓

C.1.11. Special Controls

- Items marked with an asterisk * appear on a setup screen.

Function	Size	Graphic
Brightness up	88x49	>

Function	Size	Graphic
Brightness down	88x49	‹
Bump Down* (Analog output coarse adjustment)	88x49	⩔
Bump Up* (Analog output coarse adjustment)	88x49	⩓
Clear Total*	88x49	C
Discrete I/O output off*	88x49	○
Discrete I/O output on*	88x49	●
No / Cancel	88x49	✕
Nudge Down* (Analog output fine adjustment)	88x49	∨
Nudge Up* (Analog output fine adjustment)	88x49	∧
Pause	88x49	⏸
Reset*	88x49	↻
Reset Trans. Counter	88x49	↻
Start	88x49	▶

Function	Size	Graphic
Stop/Abort	88x49	■

C.1.12. Fill PAC – Basic Automatic Filling

C.1.12.1. Basic Auto Filling – Softkeys on Start-up Screen

Function	Size	Graphic
Exit to Home Page	88x49	⌂
Application Settings	88x49	⚙
Next Page	88x49	»
Start	88x49	▶
Target Table	88x49	🗎
Login	88x49	👤

C.1.12.2. Basic Auto Filling – Application Settings Icons

Function	Size	Graphic
Target Operation	88x49	⌖
Application Settings	88x49	⚙
Target Table	88x49	🗎
Discrete Inputs	88x49	➔

Function	Size	Graphic
Discrete Outputs	88x49	⮕

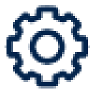
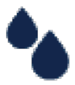
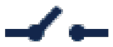
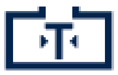
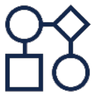
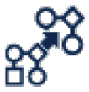
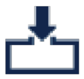
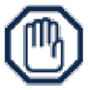
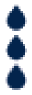
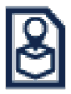
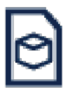
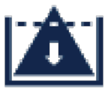
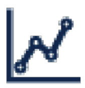
C.1.13. Fill PAC – Advanced Automatic Filling

C.1.13.1. Advanced Automatic Filling – Softkeys on Start-up Screen

Function	Size	Graphic
Exit to Home Page	88x49	⌂
Application Settings	88x49	⚙
Next Page	88x49	»
Start	88x49	▶
Material Table	88x49	🗎
Login	88x49	👤
Number of Cycles	88x49	◇

C.1.13.2. Advanced Automatic Filling –Application Settings Icons

Function	Size	Graphic
Action Log	88x49	🗒
Advanced Settings	88x49	⚙

Function	Size	Graphic
Feed Alarm	88x49	
Autospill Adjustment	88x49	
Auxiliary Output	88x49	
Container Tare	88x49	
Cycles	88x49	
Cycle Transition	88x49	
Refill	88x49	
Interlocks	88x49	
Jog	88x49	
Feed Settings	88x49	
Material Table	88x49	
Overfill Adjustment	88x49	
PAC Statistics	88x49	

Function	Size	Graphic
Timing	88x49	⏱
Tolerance Acceptance	88x49	±✓
Work Mode	88x49	↱◇ ○↲
Discrete Inputs	88x49	➔◯
Discrete Outputs	88x49	◯➔

C.1.13.3. Advanced Automatic Filling – Softkeys in Sequence

Function	Size	Graphic
Dump/Dose	88x49	⬇
Jog	88x49	💧💧💧
Manual Accept	88x49	±✓
Pause	88x49	⏸
Refill	88x49	📥
Start	88x49	▶
Stop	88x49	■

C.1.14. Fill PAC – Drum Filling

C.1.14.1. Drum Filling – Softkeys on Start-up Screen

Function	Size	Graphic
Exit to Home Page	88x49	⌂
Application Settings	88x49	⚙
Next Page	88x49	»
Start	88x49	▶
Material Table	88x49	🗎
Login	88x49	👤
Number of Cycles	88x49	○◇□

C.1.14.2. Drum Filling – Application Settings Icons

Function	Size	Graphic
Action Log	88x49	🗒
Auxiliary Output	88x49	⊸
Container Tare	88x49	T
Cycles	88x49	○◇□○

Function	Size	Graphic
Cycle Transition	88x49	◇
Discrete Inputs	88x49	➜
Discrete Outputs	88x49	➜
Drip Pan Control	88x49	💧⚙
Feed Alarm	88x49	⚙
Advanced Settings	88x49	⚙
Feed Settings	88x49	📄
Material Table	88x49	📄
Interlocks	88x49	✋
Jog	88x49	💧
Lance Control	88x49	⚙
Lance Timing	88x49	⏱
Material Table	88x49	📄

Function	Size	Graphic
Overfill Adjustment	88x49	⚠
PAC Statistics	88x49	📈
Timing	88x49	⏱
Tolerance Acceptance	88x49	±✓
Work Mode	88x49	⇄

C.1.14.3. Drum Filling – Softkeys in Sequence

Function	Size	Graphic
Jog	88x49	💧
Pause	88x49	❚❚
Start	88x49	▶
Stop	88x49	■
Tolerance Acceptance	88x49	±✓

C.1.15. Formulation PAC

Function		Graphic
Application setting	88x49	⚙

Function		Graphic
Barcode reader	88x49	
Continue unfinished recipe	88x49	
Extension	88x49	
Export recipe	88x49	
Formulation with recipe	88x49	
Formulation without recipe	88x49	
General setting	88x49	
Manual formulation	88x49	
Printing out	88x49	
Recalculating	88x49	
Recipe management	88x49	
Reset the Transaction Table	88x49	
Save recipe	88x49	
Transaction data	88x49	

Function		Graphic
Turning pages or copies	88x49	︽ ︾
View by copy	88x49	❐
View by material	88x49	⬡⬡ ⬡⬡
View next copy	88x49	⏭
View previous copy	88x49	⏮

D Loading TaskExpert Files

D.1. Loading Methods

The following procedures outline methods used to install the TaskExpert file(s) and bitmap images to the IND500x.

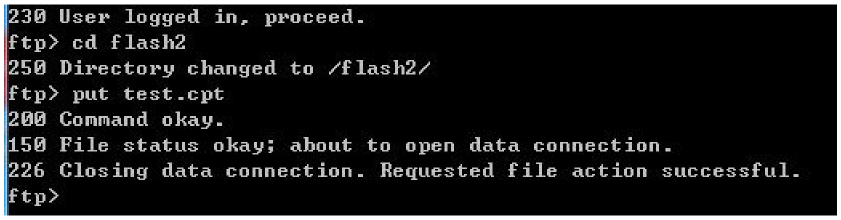
The TaskExpert file(s) and .bmp file(s) can be loaded using either FTP or Serial.

D.1.1. Loading via TaskExpert PC Tool (source code required)

1. With the project open and compiled, select the "Upload File to Terminal" icon.
2. Key in the IND500x IP address and user (default: user **Admin**, no password).
3. Select the files to be uploaded to the terminal.
4. Press Upload.

D.1.2. Loading via FTP

1. Connect the IND500x to the ACM500 and connect the ACM500 to the network.
2. Use the FTP utility to connect. The default login values are: user **Admin**, no password.
3. After login, send the command **cd flash2** to enter the flash2 folder.



```
230 User logged in, proceed.
ftp> cd flash2
250 Directory changed to /flash2/
ftp> put test.cpt
200 Command okay.
150 File status okay; about to open data connection.
226 Closing data connection. Requested file action successful.
ftp>
```

Figure D-1: IND500x FTP Server Login

4. Type **put.xxx.cpt** and press enter to upload the file, where **xxx.cpt** is the name of the file to upload.

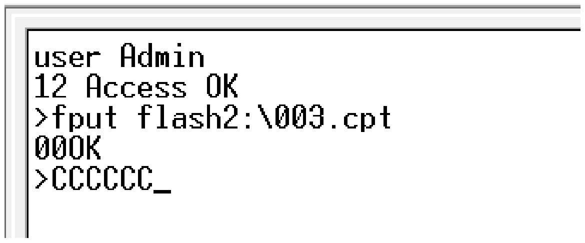
D.1.3. Loading via Serial

- The COM1 parameters set in IND500x must match the parameters set for the serial terminal program.

1. Connect to IND500x using a serial terminal program. Settings for the serial terminal program should be:

   115200 bps, 8 bits, No parity, 1 Stop bit, No flow control

2. Login to the IND500x shared data server (type: **user Admin**).



```
user Admin
12 Access OK
>fput flash2:\003.cpt
00OK
>CCCCCC_
```

Figure D-2: IND500x Shared Data Server Login

3. Use the **fput** command to notify IND500x of the flash2:\\ incoming file.

- Note: Do not enclose the filename in quotes (" ").

4. Once the "CCCC…" begins to display (Figure D-1), navigate to the **Transfer > Send File** menu in the serial terminal program.
5. Select the file to send – for example, Drive570.cpt.
6. Select the 1K Xmodem protocol.
7. Press Send. Once the file has been completely sent, the IND500x will return an <OK> back to the serial terminal program.
8. Perform this same process for each required file (e.g. **fput tempid.bmp**).

Once all the files are loaded, the application can be set to either Manual Start or Auto Start in the IND500x setup menu at **Application > TaskExpert > Start**.

E Using IND500x Fonts

E.1. Available Fonts

Eighteen configurations are available for setting fonts for display objects in the IND500x terminal. The table below includes a list of configurations, together with the font name, its height, and the number of characters that will fit on the IND500x display when that font is used.

Tool Font	Character Size, Height x Width
Alpha 8 pt	8 x 6
Alpha 12 pt	12 x 6
Alpha 16 pt	16 x 8
Chinese 12 pt	12 x 12
Chinese 16 pt	16 x 16
Num 16x9 pt	16 x 9
Num 16x10 pt	16 x 10
Num 16x12 pt	16 x 12
Num 16x14 pt	16 x 14
Num 32x15 pt	32 x 15
Num 32x17 pt	32 x 17
Num 32x20 pt	32 x 20
Num 32x24 pt	32 x 24
Num 36x20 pt	36 x 20
Num 36x23 pt	36 x 23
Num 36x30 pt	36 x 30
Num 58x32 pt	58 x 32
Num 58x43 pt	58 x43
Alpha H30	H30
Num H52	H52
Alpha H26	H36
Num H26	H26
Num H98	H98

Tool Font	Character Size, Height x Width
Num H66	H66
Num H49	H49
Num H33	H33
Alpha H16	H16
Alpha H22	H22
Alpha H26	H26
Alpha H36	H36
Alpha H42	H42
Chinese 26 pt	H26
Chinese 30 pt	H30
Num H24	H24
Num H74	H74
Num H80	H80
Num H130	H130

E.1.1. Notes

- When any alpha characters are used, such as a message on the display, the Alpha font *must* be used. If any of the Num fonts are used when alpha characters are part of the text, a blank space will appear on the IND500x display where the alpha character would be expected to appear.
- The Textbox, Combobox, and DataGrid objects only support the Alpha font.
- The Num fonts are used to display numeric characters only (including the decimal point). The reason for the various fonts is to support multiple widths of numeric characters.
- In general terms, 1 line is considered Small, 2 lines is considered Medium, and 4 lines is considered Large.

E.1.2. Examples

The following images show three examples of font in small, medium and large sizes.

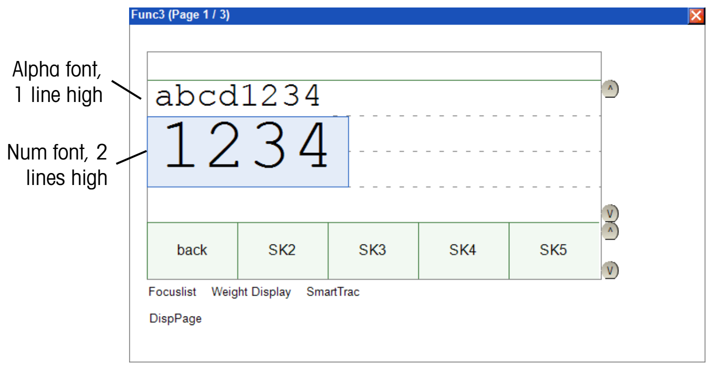


Figure E-1: Small (Alpha) and Medium (Num3) Fonts

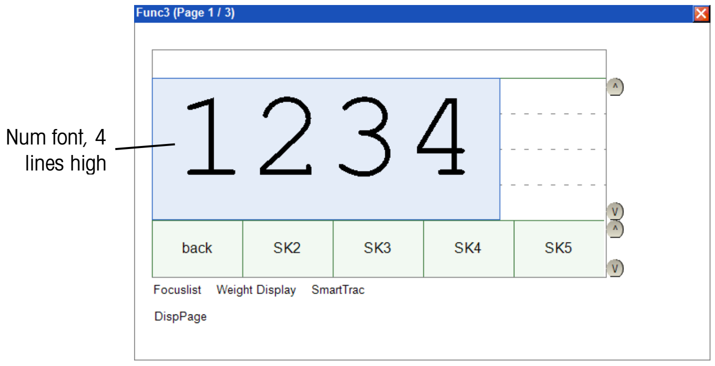


Figure E-2: Large (Num8) Font

F Release Notes and Revision History

Document Revision	Firmware Version	Date	Changes
A	–	11/2022	(First release)
B	–	09/2023	Revision
C	–	01/2024	Revision

To protect your product's future:
METTLER TOLEDO Service assures the quality, measuring accuracy and preservation of value of this product for years to come.

Please request full details about our attractive terms of service.

www.mt.com/service

www.mt.com/IND500x
For more information

Mettler-Toledo, LLC
1900 Polaris Parkway
Columbus, OH 43240
Phone 800 438 4511
Fax 614 438 4900

30753837