METTLER TOLEDO

TaskExpert™

Reference Manual

www.mt.com

64060431
(06/2011).05

Contents

Appendix G Capturing Screen Shots in the IND780 . G-1

Using Screen Capture G-1

Chapter 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.

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 **IND780 and TaskExpert** link.

Supported Devices

The following devices are supported by the current version of TaskExpert™:

Model Name	Revision
IND780	v. 6.5.15 or higher
IND560	v. 3.02 or higher

Chapter 2 Variable Expression Usage

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

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 4 types of variables in TaskExpert projects – Local, Task Global, Shared and Global.

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).

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).

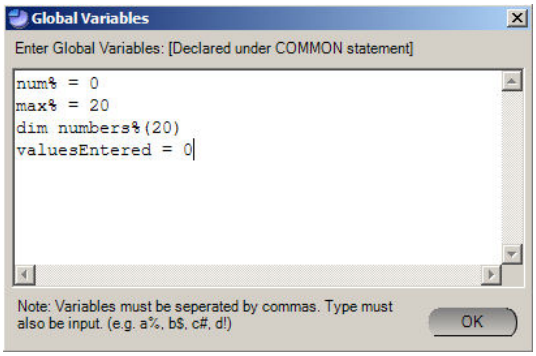


Figure 2-1: Global Variables

Shared Data variables

A shared variable can be selected from the **View I Variables I 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.

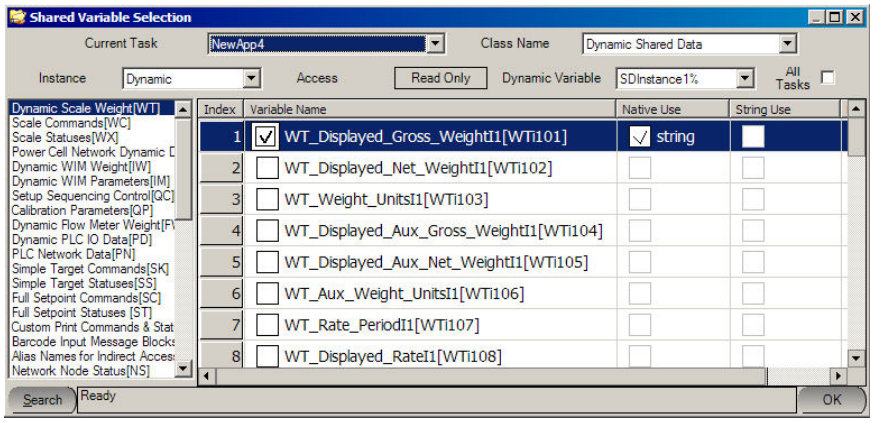


Figure 2-2: Shared Variable Selection Window

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%).

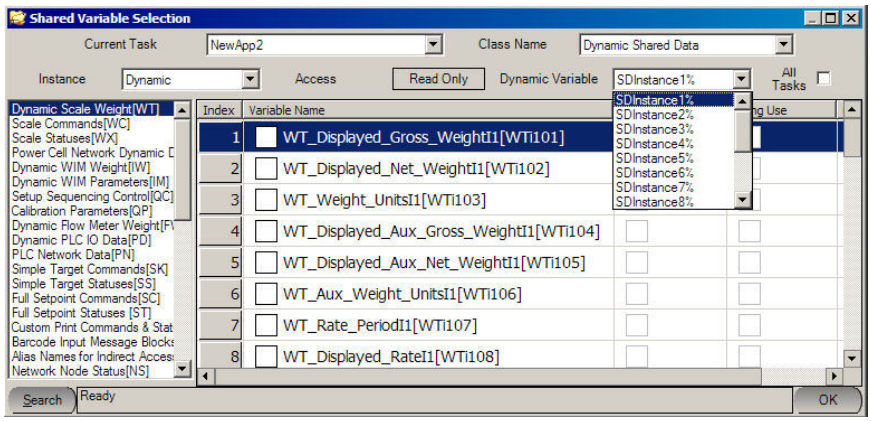


Figure 2-3: Shared Variable Selection Window, Dynamic Variable List Showing

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

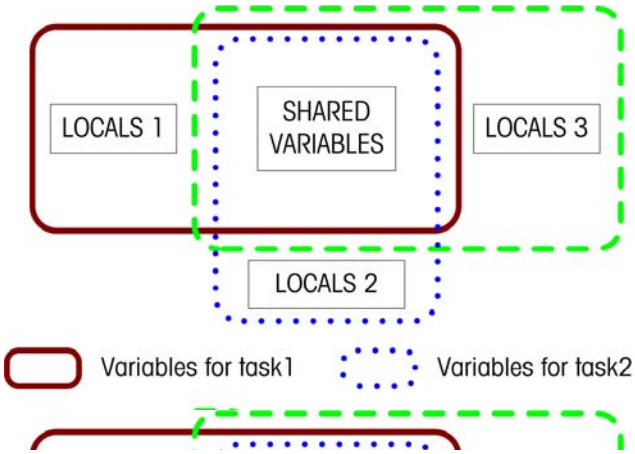


Figure 2-4: Variable Scoping in a Project

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.

Variable Usage

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 2,048 in the IND780 and 1,000 in the IND560.

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

Expression Usage

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.

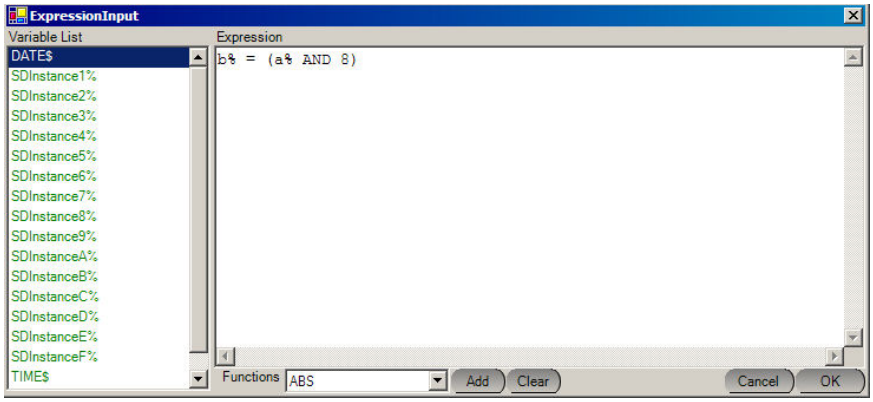


Figure 2-5: Expression Input Window

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$).

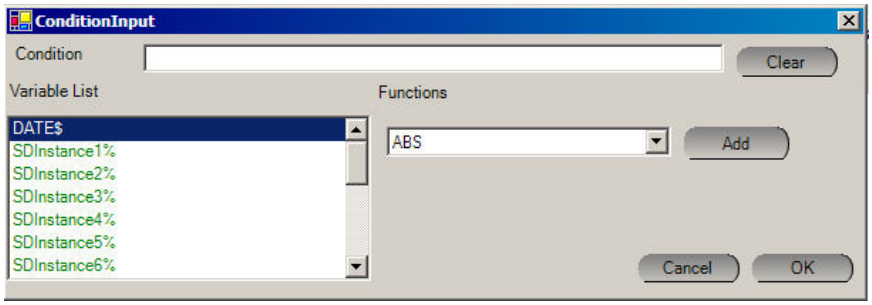


Figure 2-6: Condition Input Window

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.

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)
```

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.

Example 1:
```
A>75 OR B<20
```

Example 2:
```
B% = (A% OR C%)
```

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.

Example 1:
```
A>75 XOR B<20
```

Example 2:
```
x%=(4 XOR A%)
```

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.

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.

Syntax:

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

(numeric-expression) = any numeric expression.

Example:
ABS (45.5-100)
Value: 54.5

ATN

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

Syntax:
ATN(numeric-expression)
(numeric-expression) = any numeric expression.

Example 1:
ATN (.75)
Value (in radians): 0.6435011

Example 2:
ATN (.9)
Value (in radians): 0.7328151

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.

Syntax:
CINT(numeric-expression)
(numeric-expression) = any numeric expression.

Example 1:
CINT (12.49)
Value: 12

Example 2:
CINT (-12.50)
Value: -13

COS

Returns the cosine of a specified angle expressed in radians.

Syntax:
```
COS(angle)
```
Angle = angle expressed in radians

Example:
```
pi# = 3.141592654
COS (180 * pi# / 180)
```
Value: -1

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.

Syntax:
```
CSNG(numeric-expression)
```
(numeric-expression) = any numeric expression

Example:
```
CSNG (975.342151523497)
```
Value: 975.342152

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.

Syntax:
```
EXP(numeric-expression)
```
(numeric-expression)= any numeric expression

Example 1:
```
EXP (0)
```
Value: 1

Example 2:
```
EXP (1)
```
Value: 2.718282

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.

Syntax:
`INT(numeric-expression)`

(numeric-expression) = any numeric expression

Example 1:
`INT (12.54)`

Value: 12

Example 2:
`INT (-99.4)`

Value: -100

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.

Syntax:
`LOG(numeric-expression)`

(numeric-expression) = any positive numeric expression

Example 1:
`LOG (5)`

Value : 0.69897

Example 2:
`LOG (EXP(1))`

Value: 1

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.

Syntax:
`RANDOMIZE [seed%]`

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

Example:
```
RANDOMIZE
RANDOMIZE a%
```

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.

Syntax:
```
RND[(n#)]
```
n# = a value that sets how RND generates the next random number.

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

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.

Syntax:
```
SGN(numeric expression returns)
```
Example:
```
SGN (-15)
```
Value: -1

SIN

Returns the sine of a specified angle expressed in radians.

Syntax:
```
SIN(angle)
```
angle = angle expressed in radians

Example:
```
pi# = 3.141592654
SIN (90 * pi# / 180)
```
Value: 1

SQR

Returns the square root of a positive numeric expression.

Syntax:
```
SQR(numeric-expression)
```
(numeric-expression) = any numeric expression

Example:
```
SQR (25)
```
Value: 5

TAN

Returns the tangent of a specified angle expressed in radians.

Syntax:

```
TAN(angle)
```

angle = angle expressed in radians

Example:

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

Value: 1

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 characters for the IND780.

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.
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.

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

ASC

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

Syntax:

```
ASC(stringexpression$)
```

(stringexpression) = [String] Any string expression.

Example:

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

Output: 81

The ASCII value of a capital Q is 81.

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.

Syntax:

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

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

Example:

```
CHR$(65)
```

Output: A

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.

Refer to the example provided in **CKSUM.zip**.

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.

Example:

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

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.

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.

Example:

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

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.

Refer to the example provided in **CKSUM.zip**.

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

Syntax:

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

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

Example 1:

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

Example 2:

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

Output

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

CVI, CVS, CBD

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

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

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

DATE$

Sets or returns the terminal system date.

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.

Example

DATE$ = "2007-10-16"

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.

Syntax:

EOF(filenumber%)

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

Example:

a% = EOF(1)

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.

Syntax:

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

"event name" = name of the event

Example:

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

HEX$

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

Syntax:
```
HEX$(numeric-expression)
```
(numeric-expression) = any numeric expression.

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

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$=INKEY$
```

Syntax:
```
INKEY$
```

Example 1:
```
A$=INKEY$
```

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.

Syntax:
```
INSTR(string1$,string2$)
```
string1 = [String] String expression being searched

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

Example:
```
DIM prglst$(5)
prglst$(1)="abcdefgh"
prg$="bcd"
x$ = INSTR(prglst$(1),prg$)
```
Output: 2

JULDATE()

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

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.

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.

Example:

```
Datetime$ = "2009-01-29 09:55:23"
MyDatetime# = JulDate(Datetime$)
```

Output:

```
2009029.413466435
```

KEYSRC()

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

Syntax:

```
KEYSRC()
```

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
```

LCASE$

Converts a string to lower case.

Syntax:

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

stringexpression = [String] Any string expression.

Example:

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

Result: A$ = "good morning, sunshine"

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.

Syntax:

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

stringexpression = [String] Any string expression.

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

Example:

```
a$ = "TaskExpert – IND780"
b$ = LEFT$(a$, 10)
```

Result: b$ = "TaskExpert"

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.

Syntax:

```
LEN(stringexpression$)
```

stringexpression = [String] Any string expression.

Example:

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

Output: ABCC has length 4

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.

Syntax:

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

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

#number = The number of records.

Example: LOC(1) = 50

LOF(

Returns the length of a file.

Syntax:

```
LOF(filenumber%)
```

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

Example: size# = LOF(1)

LTRIM$

Removes the spaces from the beginning of a string.

Syntax:

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

stringexpression = [String] Any string expression.

Example:

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

Result: b$="12345"

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.

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.

Example:

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

Output: Cambridge?

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

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.

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.

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.

Example:

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

Output: 123abc789

OCT$

Converts a number to an octal string.

Syntax

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

numeric expression = Any numeric expression.

Example

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

Output: 8 decimal is 10 octal

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.

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.

Example

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

Result: b$ = ″0abc0″

PADL$

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

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.

Example:

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

Result: b$ = ”00aBc”

Example:

```
b$ = PADL $(a$, 7,”C”)
```

Result: b$ = ”CCCCaBc”

Example:

```
b$ = PADL $(a$, 3,”C”)
```

Result: b$ = ”aBc”

PADR$

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

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.

Example:

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

Result: b$ = ”aBc00”

Example:

```
b$ = PADR$(a$, 7,”C”)
```

Result: b$ = ”aBcCCCC”

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.

Syntax:

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

stringexpression = [String] Any string expression.

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

Example:

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

Output: EXPERT

RTRIM$

Removes spaces from the end of the string.

Syntax:

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

stringexpression = [String] Any string expression.

Example:

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

Result: b$ = "Hello Cambridge"

SPACE$

Returns a string of spaces. Used to indent text.

Syntax:

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

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

Example:

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

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.

Syntax:

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

numeric expression = Any numeric expression.

Example:

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

Output: 2.5

STRING$

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

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.

Example:

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

Output: -----

TIMDAT$

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

Syntax:

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

Example:

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

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

TIME$

Sets or returns the terminal system time.

Syntax:

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

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

Example

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

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.

Syntax:

```
TIMER()
```

Example:

```
time#=TIMER()
```

UCASE$

Converts a string to upper case.

Syntax:

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

Stringexpression = [String] Any string expression.

Example:

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

Result: A$ = "GOOD MORNING, SUNSHINE"

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.

Syntax:

```
VAL(stringexpression$)
```

stringexpression = [String] Any numeric string expression.

Example:

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

Output: 76

Chapter 3 Programming Control Commands

Command Reference

Note: Commands marked with an asterisk (*) refer only to the IND780.

Command	Usage
Beep*	Sounds the terminal beeper for specified number of milliseconds.
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.
FloatArray*	Formats a floating-point number and insert it into a TaskGlobal integer array.
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.
IntArray*	Formats an integer number and insert it into a TaskGlobal integer array.
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.
Read SD$*	Reads a Shared Data field, converts it to a string, and returns the string data to the application.
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.

Command	Usage
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.
WriteSD*	Writes a Shared Data field with the data formatted as a string.

Beep

IND780 only

Sounds the terminal beeper tone for the specified number of milliseconds. Used to signal an error or warn the user of the consequences of an action.

Refer to the example provided in **Switch.zip**.

Properties:

Milliseconds = [Integer] The number of milliseconds that you want the tone to sound.

Call Sub

Branches to a subroutine. Used in conjunction with RETURN.

Refer to the example provided in **FileCreation.zip**.

Properties:

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

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.

Refer to the example provided in **Chain.zip**.

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.

ChainCall

The CHAINCALL command as 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.

Refer to the example provided in **ChainCall.zip**.

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.

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.

Refer to the example provided in **ChainCall.zip**.

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.

End

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

Most of the **.zip** files contain examples of the End command.

Expression

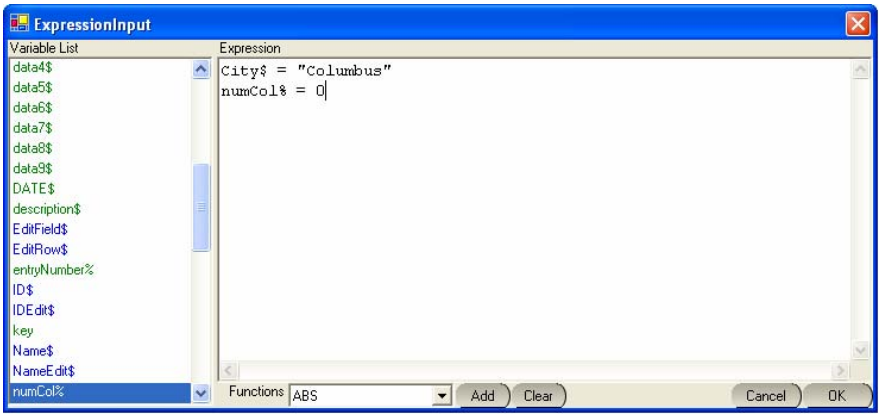
The Expression block is used to declare local variables or set expressions within an application. A variable list is included in the ExpressionInput box and updates with every new global task variable and shared data variable that is added to the application. These variables can be accessed by double-clicking 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.

Refer to the example provided in **Combobox.zip**.

Properties:

Value = Any assignment expressions



FloatArray

Programming Control — FloatArray

Format a floating-point number and insert it into a TaskGlobal integer array. Use this function in conjunction with the WriteArray function to insert a floating-point number into a Shared Data byte array in proper format.

Refer to the example provided in **FloatArray.zip**.

Properties:

Array Name	[String] Name of the TaskGlobal integer array containing the data.	
Offset	[Integer] Index location into the TaskGlobal integer array to set the formatted floating-point number.	
Var	[Double] Floating-point variable or constant.	
Status	SUCCESS	0

For

Programming Control — For

Repeats the block of statements inside the loop.

Refer to the example provided in **Switch.zip**.

Properties:

From = [Integer] Loop initial value

To = [Integer] Loop final value

Step = [Integer] Step increment

Counter Variable = [Integer, Output/Result variable] Loop counter

Goto

Branches unconditionally to a specified line.

Refer to the example provided in **EventHandling.zip**.

Properties:

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

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.

Refer to the example provided in **GetWeight.zip**.

Properties:

Condition = Conditional Expression

IntArray

Format an integer number and insert it into a TaskGlobal integer array. Use this function in conjunction with the WriteArray function to insert an integer number into a Shared Data byte array in proper format.

Refer to the example provided in **IntArray.zip**.

Properties:

Array Name	[String] Name of the TaskGlobal integer array containing the data.
Offset	[Integer] Index location into the integer array to copy the formatted integer number.
Var	[Integer] Integer variable or constant.
Length	[Integer] Optional length of the integer. The legal values are 1, 2, 3 or 4. The default value is 4.
Status	SUCCESS 0

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.

All the example **.zip** files contain examples of the Notes block.

ReadArray

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

Note: Indexing is zero (0) based

Properties:

stat% = READARRAY("shared data name","array name")

Valid combinations:

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

Invalid combinations will result in a mismatch error.

ReadSD$

IND780 only

Reads a Shared Data field, converts it to a string, and returns the string data to the application. This command only works with local Shared Data.

Properties:

SDName = [String] A valid Shared Data field name.

Returned Data:

String representation of the Shared Data Field

For more complex fields, such as structures or arrays of longs, READSD$ returns a string with a caret (^) character indicating the end of each string in the field.

ResetIND

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

Refer to the example provided in **ResetIND.zip**.

Restart

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.

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

Refer to the example provided in **Restart.zip**.

Return

Branches back to the block following the Call Sub block.

Refer to the example provided in **FileCreation.zip**.

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.

Refer to the example provided in **DIO.zip**.

Properties:

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

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.

Refer to the example provided in **Switch.zip**.

Properties:

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

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.

Refer to the example provided in **IndexedFile.zip**.

Properties:

Condition = While loop conditional expression

WriteArray

IND780 only

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

Note: Indexing is zero (0) based

Properties:

stat% = WRITEARRAY("array name", "shared data name")

Valid combinations:

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

Invalid combinations will result in a mismatch error.

WriteSD

IND780 only

Writes a Shared Data field with the data formatted as a string. The command converts the string data to the proper format of the Shared Data field before writing it. This command works only with local Shared Data.

Properties:

SDName = [String] A valid Shared Data field name.

Data = [String] A string to write to Shared Data. For more complex Shared Data Fields such as structures or arrays of longs, the application must insert caret (^) characters after each individual string in the input string.

Returned Data:

WRITESD returns the status of the operation. A non-zero status indicates that the operation has failed.

Chapter 4 Display and Keyboard Commands

Command Reference

Display Commands

Note: Except as indicated, commands are common to IND560 and IND780.

Command		Usage
IND560	**IND780**	
	AddLine	Adds a line from the previous point in an existing LineGraph object to this new point.
	AddPoint	Adds a new point in an existing LineGraph object.
Clear Application Display		Clears display objects on the application display window.
	Bargraph	Displays the contents of a Shared Data item as a bar graph in the application window.
Combobox		Displays Combobox on the screen in the application window.
DataGrid		Displays a DataGrid on the screen in the application window.
	DataView	Displays a DataView on the screen in the application window.
DispApp		Selects options for the application 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.
	FileGraph	Graphs the contents of a file as an XY line graph plotted 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
LCD Display		Turns on/off the LCD display. Adjust the display contrast. Turn off the backlight.
	Linegraph	Displays an XY line graph in the application window.
Popup		Draws a POPUP Box in the application window.

Command		Usage
IND560	**IND780**	
	Scroll Text Display	Scrolls text messages down the display.
	SDGraph	Displays the contents of a Shared Data variable(s) as a real-time XY line graph, graphing data at specific intervals.
	Set Colors	Sets the colors used by the application window, or returns the window to its default colors.
	Set Font	Changes the setting for one of the four 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.

SoftKey Commands

Command		Usage
IND560	**IND780**	
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 Pop	Pops SoftKey page from the top of stack to current active SoftKey page.
	SoftKey Push	Pushes current active SoftKey Page to top of SoftKey page stack.
SoftKey Read		Reads one SoftKey page instance and write it to another SoftKey page instance.
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.

Keyboard Commands

Command		Usage
IND560	IND780	
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.
	VCONSOLE	Interfaces to the TaskExpert application through Shared Data.

AddLine

Display & Keyboard AddLine

IND780 only

Add a line from the previous point in an existing LineGraph object to this new point.

Refer to the examples provided in **LineGraph1.zip**, **LineGraph2.zip** and **RandomPlot.zip**.

Properties:

Index [Integer] Index of an existing LineGraph object in the application display.

X-Value [Double] The value along the X-axis for the data point.

Y-Value [Double] The value along the Y-axis for the data point.

Color Select line color

Options: Black (default), Red, Blue, Green, White, Orange, Yellow, Purple

Status Refer to return statuses for SDGraph on page 4-29.

AddPoint

Display & Keyboard AddPoint

IND780 only

Add a new point in an existing LineGraph object.

Refer to the examples provided in **LineGraph1.zip**, **LineGraph2.zip** and **RandomPlot.zip**.

Properties:

Index	[Integer] Index of an existing LineGraph object in the application display.
X-Value	[Double] The value along the X-axis for the data point.
Y-Value	[Double] The value along the Y-axis for the data point.
Color	Select point color Options: Black (default), Red, Blue, Green, White, Orange, Yellow, Purple
Status	Refer to return statuses for SDGraph on page 4-29.

Clear Application Display

Clear display objects on the application display window.

Refer to the example provided in **DrawDisplay.zip.**

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

Bargraph

IND780 only

Displays the contents of a Shared Data item as a bar graph in the application window. The application window appears in the IND780 screen immediately below the Weight / SmartTrac display and above the SoftKey display. If the Shared Data field is a "callback" field, the function automatically updates the bar graph display whenever the contents of the shared data changes.

Refer to the example provided in **Bargraph.zip.**

Properties:

Index	Index of the display object in the application display. Legal values are 1 to 49. 50 is reserved for Single-Line Data Entry prompt.
x-coordinate **y-coordinate**	Position of the bar graph relative to the top left corner of the application display window.

type	Orientation of the graph, vertical or horizontal.
width	Width of the graph in pixels.
height	Height of the graph in pixels.
min	[Integer] Lowest value on the graph. The graph will start registering at this value.
max	[Integer] Highest value on the graph. The graph will stop registering at this value.
SDName	[String] Six-character name of the numeric Shared Data field. Field must be a byte, integer, long, float or double data type.
border	[Optional] Creates a black border around the bar graph.
lowTol	[Optional] Enables a low tolerance line on the bar graph. A line is shown on the graph at this value. Values below this will be shown in Color1, values above it in Color2.
highTol	[Optional] Enables a high tolerance line on the bar graph. A line is shown on the graph at this value. Values above this line will be shown in Color3.
color1	Base color black, red, blue, green, white, orange, yellow, purple. Default is blue. If no tolerances are specified, this will set the color of the bar graph. If tolerances are specified, this will set the color for values below lowTol.
color2	Color for values above lowTol black, red, blue, green, white, orange, yellow, purple. Default is green.
color3	Color for values above highTol black, red, blue, green, white, orange, yellow, purple. Default is red.

Combobox

Displays the Combo Box on the screen in the application window. The application window appears in the IND780 screen 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.

Refer to the example provided in **Combobox.zip**.

Properties:

Index	[Integer] The index of the display object in the application display. Legal values are 1 to 49. Object 50 is reserved for Single-Line Data Entry Line Prompt. **Note**:The IND560 supports only 20 display objects – indexes 1-19 are valid. Index 20 is reserved for the Data Entry Line.
Index Variable Instance	[Integer constant only, Optional] Index of the shared variables to be associated. Legal values are 1-49.
X-Coordinate	[Integer] Position of the text display relative to the top, left-hand corner of the application display window. Legal values = 0 to 320.
Y-Coordinate	[Integer] Position of the text display relative to the top, left-hand corner of the application display window. Legal values = 0 to 240.
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. Select font from list: Arial (default) and Courier New in 10, 12, 14 and 16 point sizes. The application can change the size of individual font using the SetFont function. IND560 font options are Alpha, Chinese, Num1 and Num5. Please refer to **Appendix E**.
Color	[Optional] Select text color. Options: Black (default), Red, Blue, Green, White, Orange, Yellow, Purple.
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 tx0150. 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.

Console

Note: In the IND780, the CONSOLE is set to PUBLIC by default when a TaskExpert application is running. In the IND560, 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.

Refer to the example provided in **Console.zip**.

Properties:

Note: Properties marked with an asterisk (*) are available for the IND780 only.

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.

Font* [Optional] Font type and size of the display.

Select font from list: Arial (default) and Courier New in 10, 12, 14 and 16 point sizes.

The application can change the size of individual font using the SetFont function.

Color* [Optional] Select text color.

Options: Black (default), Red, Blue, Green, White, Orange, Yellow, Purple.

* These properties are available only in the IND780.

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.

Refer to the example provided in **DataGrid.zip**

Properties:

Note: Properties marked with an asterisk (*) are available for the IND780 only.

Index [Integer] Index of the display object in the application display. Legal values are 1 to 49. Object 50 is reserved for Single-Line Data Entry Line Prompt.

Note:The IND560 supports only 20 display objects – indexes 1-19 are valid. Index 20 is reserved for the Data Entry Line.

Index Variable Instance [Integer, Optional] Index of the shared variable to be associated. Legal values are 1 to 49.

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, IND780 only:

"SELECT * FROM A0 WHERE ID = 9"

"SELECT ID,shortId,description,data1,data2,data3 FROM A1"

If the SQL Query does not contain a "where" clause, then DataGrid makes all the records in the table available for display.

Note: The SQL Query must reference column "ID" first when any editing is allowed.

Examples, IND560 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] Comma-separated of columns that TaskExpert displays in the DataGrid that the operator may edit. For example, "3" enables the operator to edit column 3 of the table.

Font

[Optional] Font type and size of the display.

Select font from list: Arial (default) and Courier New in 10, 12, 14 and 16 point sizes.

The application can change the size of individual font using the SetFont function.

IND560 font options are Alpha, Chinese, Num1 and Num5. Please refer to **Appendix E**.

Color*

[Optional] Select text color.

Options: Black (default), Red, Blue, Green, White, Orange, Yellow, Purple.

Title

Specifies the text displayed above the data grid. If not specified, no title will be shown.

TitleFont	Type and size of font used to display the title text. Select font from list: Arial (default) and Courier New in 10, 12, 14 and 16 point sizes. The application can change the settings of individual fonts using the SetFont function.
Height [IND780]	[Integer] The total height of the datagrid. If the height is not specified, the datagrid uses all of the space between the status bar and the softkeys with the weight display and SmartTrac turned off.
Y-Coordinate [IND780]	[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:

SUCCESS	0
ERR_TOO_MANY_DISPLAY_OBJECTS	-1
ERR_INVALID_HANDLE	-3
ERR_OBJECT_NOT_ALLOCATED	-4
ERR_CANNOT_SCROLL	-6
ERR_DISPLAY_UNICODE_CONVERSION	-7
ERR_INVALID_TASK_EXPERT_INSTANCE	-9
ERR_INVALID_TYPE_ARGUMENT	-10
ERR_CANNOT_ACCESS_DATABASE_TABLE	-11

Edited Value	[Optional] Edited result, mapped to shared variable tx0151.
RowID	[Optional] Edited row ID, mapped to shared variable tx0152.
ColID	[Optional] Edited column ID, mapped to shared variable tx0153.
Selected Row	[Optional] When the operator selects a new row, the DataGrid writes the newly selected row's index to tx0154.

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 tx0150. 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.

If the operator edits a DataGrid field, DataGrid returns the new contents of the edited field to the TaskExpert application in Shared Data field in tx0151, the shortID$ of the edited row in tx0152, the column number in tx0153, and row the index in tx0154.

Data Entry Validation:

The application may have an event service routine that enables the application to validate or invalidate the edited data.

The application can validate the entry to DataGrid by setting tx0155 = "Accept", or can invalidate the entry by setting tx0155 to a rejection message. DataGrid "pops up" the rejection message to the operator. If the application validates the entry, DataGrid enters the data into the DataBase Table.

If DataGrid does not receive a response from the application within five seconds, DataGrid automatically accepts the edited entry and enters it into the Database Table.

DataView

IND780 only

Display a DataView on the screen in the application window. The DataView displays columns of a custom database table for the operator.

Refer to the examples provided in **DataView.zip**.

Properties:

Index [Integer] Index of the display object in the application display. Legal values are 1 to 49. Object 50 is reserved for the Single-Line Data Entry Line Prompt.

File Name [String] The full path to the database file.
Example:

```
\Storage
Card\taskexpert\data\custom.sdf
```

SQL Query [String] A valid SQL statement.
Example:

```
SELECT
LastName,FirstName,Address,City,State,Zip
FROM Employees
```

Note 1: The DataView object requires that each column be explicitly defined in the SQL Query statement. The asterisk (*) cannot be used as a "select all" in the select clause.

Note 2: If the SQL Query does not contain a "where" clause then DataView makes all the records in the table available for display.

Header List [String] Comma-separated list of column headers from the Database

Table that TaskExpert displays in the DataView For example, the Header List "Last,First,Address,City,State,Zip" defines the 6 columns from the table that will be shown in the DataView.

Column Width [String] Comma separated list of the column widths displayed. For example, "60,60,130,80,60,50" displays the LastName column in 60 pixels, FirstName column in 60 pixels, Address in 130 pixels, City in 80 pixels, State in 60 pixels and the Zip column in 50 pixels. Note: A width of 0 hides a column.

Font Font Type and Size.

Select font from list: Arial (default) and Courier New in 10, 12, 14 and 16 point sizes. The application can change the settings of individual fonts using the SetFont function.

Color Select text color.

Options: Black (default), Red, Blue, Green, White, Orange, Yellow, Purple

Title [String] Specifes the text displayed above the dataview. It this is not specified, no title will be shown.

Title Font Type and size of font used to display the title text.

Select font from list: Arial (default) and Courier New in 10, 12, 14 and 16 point sizes. The application can change the settings of individual fonts using the SetFont function.

Height [Integer] The total height of the dataview. If the height is not specified, the dataview uses all of the space between the status bar and the softkeys with the weight display and SmartTrac turned off.

Y-Coordinate [Integer] The position of the dataview relative to the top of the application display window.

Selected Row [Optional] When the operator selects a new row, DataView writes the newly selected row's first data column content to tx0154.

Status

SUCCESS	0
ERR_TOO_MANY_DISPLAY_OBJECTS	-1
ERR_INVALID_HANDLE	-3
ERR_OBJECT_NOT_ALLOCATED	-4
ERR_CANNOT_SCROLL	-6
ERR_DISPLAY_UNICODE_CONVERSION	-7
ERR_INVALID_TASK_EXPERT_INSTANCE	-9
ERR_INVALID_TYPE_ARGUMENT	-10
ERR_CANNOT_ACCESS_DATABASE_TABLE	-11

DispApp

Display Objects → DispApp

IND560 only

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

Properties:

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

Draw Image

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

The application window appears in the IND780 screen immediately below the Weight/SmartTrac display and above the SoftKey display. There can be up to 50 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.

Refer to the example provided in **DrawDisplay.zip**.

Properties:

Index	[Integer] Index of the display object in the application display. Legal values are 1 to 49. Object 50 is reserved for Single-Line Data Entry Line Prompt. **Note**: The IND560 supports only 20 display objects – indexes 1-19 are valid. Index 20 is reserved for the Data Entry Line.
X-Coordinate	[Integer] X-Coordinate is the position of the image display relative to the top, left-hand corner of the application display window. Legal values are 0 to 320.
Y-Coordinate	[Integer] Y-Coordinate is the position of the image display relative to the top, left-hand corner of the application display window. Legal values are 0 to 240.
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 on the IND780 Compact Flash where the bitmap of the graphic display image resides.
LocalFocusedFilename	[Optional] Image file name to display when the image is focused. File must be available in Local machine.
Focused Filename	[String, Optional] This is an optional parameter that identifies the name of the file on the Compact Flash where the bitmap of the "focused" graphic display image resides. When the user identifies this optional parameter, the TaskExpert Interpreter displays this graphical image when this object is focused on the display, and displays the File Name image when the object is not focused.

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

Draw Label

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

The application window appears in the IND780 screen immediately below the Weight/SmartTrac display and above the SoftKey display. There can be up to 50 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.

Text overwrites the graphics when there is a conflict in the display space. DRAW IMAGE could be used to draw a box using a bitmap and then the DRAW TEXT command to write text inside it.

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

Refer to the example provided in **DrawDisplay.zip**.

Properties:

Index [Integer] Index of the display object in the application display. Legal values are 1 to 49. Object 50 is reserved for Data Entry Line Prompt.

Note: The IND560 supports only 20 display objects – indexes 1-19 are valid. Index 20 is reserved for the Data Entry Line.

X-Coordinate [Integer] X-Coordinate is the position of the text display relative to the top, left-hand corner of the application display window. Legal values are 0 to 320.

Y-Coordinate [Integer] Y-Coordinate is the position of the text display relative to the top, left-hand corner of the application display window. Legal values are 0 to 240.

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.

Select font from list: Arial (default) and Courier New in 10, 12, 14 and 16 point sizes.

The application can change the size of individual font using the SetFont function.

IND560 font options are Alpha, Chinese, and Num1 through Num8. Please refer to **Appendix E**.

Color [Optional] Select text color.

Options: Black (default), Red, Blue, Green, White, Orange, Yellow, Purple.

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

Draw Shared Data

Display the contents of Shared Data in the application window. The application window appears in the IND780 screen 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.

Refer to the example provided in **DrawDisplay.zip**.

Properties:

Index [Integer] Index of the display object in the application display. Legal values are 1 to 49. Object 50 is reserved for Single-Line Data Entry Line Prompt.

Note: The IND560 supports only 20 display objects – indexes 1-19 are valid. Index 20 is reserved for the Data Entry Line.

X-Coordinate [Integer] X-Coordinate is the position of the text display relative to the top, left-hand corner of the application display window. Legal values are 0 to 320.

Y-Coordinate [Integer] Y-Coordinate is the position of the text display relative to the top, left-hand corner of the application display window. Legal values are 0 to 240.

SD Name Six-character name of the Shared Data field.

Font [Optional] Font type and size of the display.

Select font from list: Arial (default) and Courier New in 10, 12, 14 and 16 point sizes.

The application can change the size of individual font using the SetFont function.

IND560 font options are Alpha, Chinese, and Num1 through Num8. Please refer to **Appendix E**.

Color [Optional] Select text color.

Options: Black (default), Red, Blue, Green, White, Orange, Yellow, Purple.

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

ExPopup

IND780 only

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 49. Object 50 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 tx0150, 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.

FileGraph

IND780 only

Graph the contents of a file as an XY line graph plotted in the application window. The file is a ".csv" file that contains the graph data. The file may define one or more lines in the line graph. A PC program, TaskExpert program, or Excel spreadsheet can create the file. The vertical axis displays the low values starting at the bottom and growing to the top as the value increases. The horizontal axis will display the low values starting at the left and growing to the right as the value increases. The graph appears in the application window, which is immediately below the Weight/SmartTrac display and above the Softkey display in the IND780 screen.

Note: The maximum number of objects that can be added to a FileGraph object is 500.

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

Refer to the examples provided in **FileGraph1.zip** and **FileGraph2.zip**.

Properties:

Index [Integer] Index of the display object in the application display. Legal values are 1 to 49. TE reserves Object 50 for the Single-Line Data Entry Line Prompt.

X-Coordinate [Integer] Position of the line graph display relative to the top, left-hand corner of the application display window.

Legal values = 0 to 320.

Y-Coordinate [Integer] Position of the line graph display relative to the top, left-hand corner of the application display window.

Legal values = 0 to 240

Width [Integer] The width of the graph in pixels.

Height [Integer] The height of the graph in pixels.

Graph Type [Integer] Select the type of graph

0 = FileGraph plots lines between successive data points as a line graph,

1 = FileGraph plots the data only as a point (or scatter) graph.

File Name [String] Path + Name of a ".csv" data file that contains a three-column, multi-row array.

- The first row of the array contains the MinX and MinY.
- The second row of the array contains the MaxX and MaxY.
- The rest of the file contains the graph points. Each row represents one point in the graph. The row must contain the X-axis data in the first column, Y-axis data in the second column, and color in the third column.
- The keyword "NEW" in the file starts a new line that FileGraph draws on the same graph.

Border Select Yes to place a black border around the line graph.

AxisX [Integer] The number of evenly spaced registration ticks to place along the X-axis.

AxisY [Integer] The number of evenly spaced registration ticks to place along the Y-axis.

Status Refer to return statuses for SDGraph on page 4-29.

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

Refer to the example provided in **Exsoftkey.zip**.

Properties:

Soft Key ID	[Integer] Index into the softkeys in the working page. The legal values are 1 to 19. Indexes 1 – 4 refer to the application keys A1 – A4. Indexes 5 -19 refer to SoftKeys 1 – 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, MOVE DOWN, and DELETE.
Key ID	[Integer] Key identifier that the SoftKey Manager passes a SoftKey to an application. It uniquely identifies the key.
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.

Focusitem

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

Refer to the example provided in **Focus.zip**.

Properties:

Status	[Integer variable, Output/Result variable, Optional] Return value of the function block. Legal values are 1 to 49. Object 50 is reserved. 0 (zero) signals an error or that nothing has the focus

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.

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

Refer to the example provided in **Focus.zip**.

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_TOO_MANY_DISPLAY_OBJECTS	-1
ERR_OBJECT_NOT_ALLOCATED	-4

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.

Refer to the example provided in **Focus.zip**.

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

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.

Refer to the example provided in **Format.zip**.

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.

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.

Refer to the example provided in **KeyCaptureSample.zip**.

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.

Refer to the example provided in **KeyCaptureSample.zip**.

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.

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 50 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.

Refer to the example provided in **ImageSD.zip**.

Properties:

Index	[Integer] The index of the display object in the application display. Legal values are 1 to 49. Object 50 is reserved for Single-Line Data Entry Line Prompt. **Note**: The IND560 supports only 20 display objects – indexes 1-19 are valid. Index 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. Legal values are 1 to 320.
Y-Coordinate	[Integer] The position of the image display relative to the top left-hand corner of the application display window. Legal values are 1 to 240.
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

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.

Refer to the example provided in **KeyCaptureSample.zip**.

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.

KeyEntry

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.

Refer to the example provided in **KeyEntry.zip**.

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.

Select font from list: Arial (default) and Courier New in 10, 12, 14 and 16 point sizes.

The application can change the size of individual font using the SetFont function.

IND560 font options are Alpha, Chinese, Num1 and Num5. Please refer to **Appendix E**.

Color [Optional] Select text color.

Options: Black (default), Red, Blue, Green, White, Orange, Yellow, Purple.

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.

Keysrc

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

Refer to the example provided in **Keysrc.zip**.

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

LCD Display

IND780 only

Turn on/off the LCD display. Adjust the display contrast (monochrome display only). Turn off the backlight.

Refer to the example provided in **LCD.zip**.

Properties:

Option	ON	Turns on the LCD and Backlight.
	OFF	Turns off the LCD.
	UP	Adjusts the contrast up for the specified number of counts.
	DOWN	Adjusts the contrast down for the specified number of counts.
	RESET	Resets LCD display
	BACKOFF	Turns the backlight off.
Up/Down Counts	[Integer, Optional] Contrast adjust value.	

LineGraph

IND780 only

Display an XY line graph in the application window. This command creates the LineGraph object, and then the TaskExpert application adds the graph points and lines using successive AddPoint and AddLine functions. The vertical axis displays the low values starting at the bottom and growing to the top as the value increases. The horizontal axis will display the low values starting at the left and growing to the right as the value increases. The graph appears in the application window, which is immediately below the Weight/SmartTrac display and above the Softkey display in the IND780 screen.

Note: The maximum number of objects that can be added to a LineGraph object is 250.

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

Refer to the examples provided in **LineGraph1.zip**, **LineGraph2.zip** and **RandomPlot.zip**.

Properties:

Index	[Integer] Index of the display object in the application display. Legal values are 1 to 49. TE reserves Object 50 for the Single-Line Data Entry Line Prompt.

X-Coordinate [Integer] Position of the line graph display relative to the top, left-hand

corner of the application display window.

Legal values = 0 to 320.

Y-Coordinate [Integer] Position of the line graph display relative to the top, left-hand corner of the application display window.

Legal values = 0 to 240.

Width [Integer] The width of the graph in pixels.

Height [Integer] The height of the graph in pixels.

MaxX [Double] The highest value that LineGraph records in the array for the X-axis. LineGraph will not register data points above this value.

MaxY [Double] The highest value that LineGraph records in the array for the Y-axis. LineGraph will not register data points above this value.

MinX [Double] The lowest value of the data that LineGraph records in the array for the X-axis. LineGraph will not graph data points below this value.

MinY [Double] The lowest value of the data that LineGraph records in the array for the Y-axis. LineGraph will not graph data points below this value.

Border Select Yes to place a black border around the line graph.

AxisX [Integer] The number of evenly spaced registration ticks to place along the X-axis.

AxisY [Integer] The number of evenly spaced registration ticks to place along the Y-axis.

Color Select line/point color

Options: Black (default), Red, Blue, Green, White, Orange, Yellow, Purple

Status Refer to return statuses for SDGraph on page 4-29.

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.

Refer to the example provided in **Messages.zip**.

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".

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

Reset Console Keys

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

Refer to the example provided in **KeyCaptureSample.zip**.

Scroll Text Display

IND780 only

The Scroll-Text Display function allows text messages to be scrolled down the display. Each time a new text message is displayed in Scroll-Text Mode, the TaskExpert scrolls the old text messages down the display to make room for the new message. The oldest message may scroll off the bottom of the application display. Scroll-Text Mode is mutually exclusive from the Static-Position display mode used in other application commands where fixed x- and y-coordinates are given. That is, it is not possible to display messages in Scroll-Text Mode and Static-Position Mode at the same time.

Refer to the example provided in **Scroll.zip**.

Properties:

Text [String] Text of the message.

Font [Integer, Optional] Font size of the display.

Enter an integer representing the font size in points – e.g., 12.

Color [Integer, Optional] Select text color.

Options: Black (default), Red, Blue, Green, White, Orange, Yellow, Purple.

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
ERR_INVALID_TYPE_ARGUMENT	-10
ERR_CANNOT_ACCESS_DATABASE_TABLE	-11
ERR_CP_HAS_TAKEN_CONTROL_OF_SOFT_KEYS	-12

SDGraph

IND780 only

Display the contents of a Shared Data variable(s) as a real-time XY line graph, graphing data at specific intervals. The graphing interval is either a regular time interval or a new occurrence of a Shared Data trigger. The vertical Y-axis displays the low values starting at the bottom and growing to the top as the value increases. The horizontal X-axis will display the low values starting at the left and growing to the right as the value increases. The graph appears in the application window, which is immediately below the Weight / SmartTrac display and above the Softkey display in the IND780 screen.

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

Refer to the examples provided in **SDGraph1.zip** and **SDGraph2.zip**.

Properties:

Index [Integer] Index of the display object in the application display. Legal values are 1 to 49. TE reserves Object 50 for the Single-Line Data Entry Line Prompt.

X-Coordinate [Integer] Position of the line graph display relative to the top, left-hand

corner of the application display window.

Legal values = 0 to 320.

Y-Coordinate [Integer] Position of the line graph display relative to the top, left-hand corner of the application display window.

Legal values = 0 to 240.

Graph Type [Integer] Select type of graph:

Line (non-scrolling)

0 = SDGraph plots lines between successive data points as a line graph, graphing a new line when one of the Shared Data variables or counts changes, and stopping when it reaches the maximum time count or trigger count value of the graph.

Scatter (non-scrolling)

1 = SDGraph plots the data as a point (or scatter) graph, graphing a new data point when one of the Shared Data variables or counts changes, and stopping when it reaches the maximum time count or trigger count value of the graph.

Line (scrolling)

2 = SDGraph plots lines between successive data points as a line graph, graphing a new line when one of the Shared Data variables or counts changes, and automatically scrolling the graphing window along the time or trigger count axis after reaching the maximum count.

Scatter (scrolling)

3 = SDGraph plots the data as a point (or scatter) graph, graphing a new data point when one of the Shared Data variables or counts changes, and automatically scrolling the graphing window along the time or trigger count axis after reaching the maximum count.

Note: For the scrolling graph types, a trigger must be defined to indicate when to scroll. This is defined with the words "time" or "trigger" for either SDNameX or SDNameY.

Width [Integer] The width of the graph in pixels.

Height [Integer] The height of the graph in pixels.

SDNameX Represents the value plotted on the X-axis.

- If SDNameX is a SharedData field name, the field must be a byte, integer, long, float or double data type. If the SDNameX is preceded by a "+", then SDGraph plots the absolute value of data value, (e.g., "+wt0114").
- If SDNameX is the word "time", the SDGraph plots a new value at each time interval, where the values on the X-axis represents the time.
- If SDNameX is the word "trigger", SDGraph plots a new value at each occurrence of a callback on the SDNameY Shared Data variable. The value that SDGraph plots on the X-axis is the occurrence number of the callback.

SDNameY Represents the value plotted on the Y-axis.

- If SDNameY is a SharedData field name, the field must be a byte, integer, long, float or double data type. If the SDName is preceded by a "+", then SDGraph plots the absolute value of data value, (e.g., "+wt0114").
- If SDNameY is the word "time", the SDGraph plots a new value at each time interval, where the values on the X-axis represents the time.
- If SDNameY is the word "trigger", SDGraph plots a new value at each occurrence of a callback on the SDNameY Shared Data variable. The value that SDGraph plots on the X-axis is the occurrence number of the callback.

MaxX [Double] The maximum value that SDGraph records along the X-axis.

- If SDNameX is "time", this value is in quarter-seconds.
- If SDNameX is "trigger", this value is the number of occurrences of SDNameY callbacks that SDGraph will plot.
- If the graph type is 0 or 1 AND SDNameX is "time" or "trigger", SDGraph will stop graphing when the count reaches MaxX.
- If the graph type is 2 or 3 AND SDNameX is "time" or "trigger", SDGraph will automatically begin scrolling the graphing window when the count reaches MaxX.

MaxY [Double] The maximum value that SDGraph records along the Y-axis.

- If SDNameY is "time", this value is in quarter-seconds.
- If SDNameY is "trigger", this value is the number of occurrences of SDNameX callbacks that SDGraph will plot.
- If the graph type is 0 or 1 AND SDNameY is "time" or "trigger" SDGraph will stop graphing when the count reaches MaxY.
- If the graph type is 2 or 3 AND SDNameY is "time" or "trigger" SDGraph will automatically begin scrolling the graph when the count reaches MaxY.

MinX [Double] The lowest value of the data that SDGraph records in the array for the X-axis. SDGraph will not graph data points below this value. The default value is zero.

- If SDNameX is "time", this value is quarter-seconds.
- If SDNameX is "trigger", this value is the number of occurrences of SDNameY callbacks that must occur before SDGraph will begin to plot.

MinY Double] The lowest value of the data that SDGraph records in the array for the Y-axis. SDGraph will not graph data points below this value. The default value is zero.

- If SDNameY is "time", this value is quarter-seconds.
- If SDNameY is "trigger", this value is the number of occurrences of SDNameX callbacks that must occur before SDGraph will begin to plot.

Time Interval [Integer] Time interval in quarter-seconds.

- When an SDName is NOT "trigger", SDGraph plots data onto the graph at these successive time intervals.
- When an SDName is "trigger", SDGraph will plot points no more frequently than this time interval.

Plot Trigger The name of a Shared Data trigger variable that can control the graphing function.

- If the SDGraph function call does NOT specify a Plot Trigger, SDGraph begins graphing normally.
- If the SDGraph function call specifies a Plot Trigger, SDGraph uses it as follows:
  1. When Plot Trigger = 1, SDGraph turns on the graphing function.
  2. When Plot Trigger transitions from 1 to 0, SDGraph stops graphing.
  3. When the Plot Trigger transitions from 0 to 1, SDGraph erases the current graph and starts a new graph.
  4. SDGraph repeats this sequence as long as the function remains active.

Border Select Yes to place a black border around the line graph.

Axis X [Integer] The number of evenly spaced registration ticks to place along the X-axis.

Axis & [Integer] The number of evenly spaced registration ticks to place along the Y-axis.

Color Select line/point color

Options: Black (default), Red, Blue, Green, White, Orange, Yellow, Purple

Status

```
TE_SUCCESS = 0,
TE_SUCCESS_SHARED_DATAPREV_USED = 1,
TE_ERR_TOO_MANY_DISPLAY_OBJECTS= -1,
TE_ERR_CANNOT_ACCESS_FILE = -2,
TE_ERR_INVALID_HANDLE = -3,
TE_ERR_OBJECT_NOT_ALLOCATED = -4,
TE_ERR_INVALID_SHARED_DATA = -5,
TE_ERR_CANNOT_SCROLL = -6,
TE_ERR_DISPLAY_UNICODE_CONVERSION = -7,
TE_ERR_XY_OUT_OF_RANGE = -8,
TE_ERR_INVALID_TASK_EXPERT_INSTANCE = -9,
TE_ERR_INVALID_TYPE_ARGUMENT = -10,
TE_ERR_CANNOT_ACCESS_DATABASE_TABLE = -11,
TE_ERR_CP_HAS_TAKEN_CONTROL_OF_SOFT_KEYS = -12,
TE_ERR_PATH_NOT_FOUND = -13,
TE_ERR_ACCESS_DENIED = -14,
TE_FILE_TOO_LARGE = -15,
```

Set Colors

IND780 only

Set the colors used by the application window, or return the window to its default colors. The application window appears immediately below the Weight/SmartTrac display and above the Softkey display. Calling SetColors with no parameters restores the default color scheme.

Each of the color parameters should be one of the following: black, red, blue, green, white, orange, yellow, purple.

Properties:

backColor	Color used for the background.
defaultForeColor	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.
borderColor	Color used for the border of display objects.

Set Font

IND780 only

Change the setting for one of the four application display fonts. This command only makes the font setting. You select which font you want to use in the application display commands.

Refer to the example provided in **SetFont.zip**.

Properties:

Index	[Integer value only] Available font index. The legal values are 1 to 8.
Font Height	[Integer] New setting for the font height – e.g., 12.
Bold	[Optional] Select from list: Normal (default) or Bold.
Font Type	[Optional] Select from list: Arial (default) or Courier New. Arial (default) defines a proportional-character-size "Arial" font. The Arial font takes less space. Courier New defines a fixed-character-size "Courier New" font. The Courier New is useful when you need to line up columns of data with fixed sizes.
Status	[Integer variable, Output/Result variable, Optional] Return value of this function block.

SmartTrac

Sets the options for the SmartTrac display

Refer to the example provided in **Smarttrac.zip**.

Properties:

Setpoint [Integer, Optional] Number of the setpoint that drives the SmartTrac display.

Display [Optional] Turns on or off the SmartTrac display. The INVISIBLE option turns off the SmartTrac display but holds its position on the display.

Options: ON, OFF, INVISIBLE

Size [Optional] Sets the size of the SmartTrac display.

Options: SMALL, MEDIUM, LARGE

SmartTrac Type [Optional] Sets the type of display.

Options: BAR GRAPH, CROSS HAIRS, THREE ZONE

Soft Key Clear

Clear the application SoftKey working page.

Refer to the example provided in **SoftKey.zip**.

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.

Soft Key Disable

Disable the SoftKeys.

Refer to the example provided in **SoftKeyControl.zip**.

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.

Soft Key Enable

Enable the SoftKeys.

Refer to the example provided in **SoftKeyControl.zip**.

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.

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.

Refer to the example provided in **SoftKey.zip**.

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.

Soft Key Pop

IND780 only

Pop SoftKey page from the top of stack to current active SoftKey page. This operation restores the state of the SoftKey page and console display to its state before the application modified it.

Refer to the example provided in **SoftKey.zip**.

Properties:

Persistence [Optional] Normally, the TaskExpert interpreter sets the SoftKey stack back to its home position when the TaskExpert program terminates. The PERSIST parameter causes the stack to be left in its current state when the program terminates. The last execution of the SKPUSH or SKPOP command before program termination determines the PERSIST state.

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.

Soft Key Push

IND780 only

Push current active SoftKey Page to top of SoftKey page stack so that the application can retrieve it for later operation. Move application-working page to current active SoftKey page for display on the console.

Refer to the example provided in **SoftKey.zip**.

Properties:

Persistence [Optional] Normally, the TaskExpert interpreter sets the SoftKey stack back to its home position when the TaskExpert program terminates. The PERSIST parameter causes the stack to be left in its current state when the program terminates. The last execution of the SKPUSH or SKPOP command before program termination determines the PERSIST state.

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.

Soft Key Read

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

Refer to the example provided in **SoftKeyRead.zip**.

Properties:

Source Entry [Integer, Optional] The number of the source SoftKey page instance. Legal values are 12 to 19.

Destination Entry [Integer, Optional] The number of the destination SoftKey page instance. Legal values are 12 to 19.

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 current top 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.

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 current displayed page.
- Instances 2 through 10 are the SoftKey processing stack.
- Instance 11 is the application-working page.
- Instances 9 through 19 are TaskExpert application temporary SoftKey storage pages.
- Instance 20 is the Home SoftKey page.

Soft Key Replace

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

Refer to the example provided in **SoftKey.zip**.

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.

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.

Refer to the example provided in **SoftKey.zip**.

Properties:

Soft Key ID [Integer] Index into the softkeys in the working page. The legal values are 1 to 19. Indexes 1 – 4 refer to the application keys A1 – A4. Indexes 5 -19 refer to SoftKeys 1 – 15.

Mode Mode of adding the SoftKey to the working page. Legal values are OVERWRITE, INSERT, 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.

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, Optional] 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, or application terminates with an error.

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.

Refer to the example provided in **Messages.zip**.

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.

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.

Refer to the example provided in **KeyEntry.zip**.

Textbox

Display the text entry box on the screen in the application window. The application window appears in the IND780 screen 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.

Refer to the example provided in **Textbox.zip**.

Properties:

Index	[Integer] Index of the display object in the application display. Legal values are 1 to 49. Object 50 is reserved for Single-Line Data Entry Line Prompt. **Note**: The IND560 supports only 20 display objects – indexes 1-19 are valid. Index 20 is reserved for the Data Entry Line.
Index Variable Instance	[Integer, Optional] Index of the shared variable to be associated. Legal values are 1 to 49.
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. Legal values are 0 to 320.
Y-Coordinate	Position of the text display relative to the top, left-hand corner of the application display window. Legal values are 0 to 240.
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.

In numeric data entry mode, "%nn" is the format specification where nn is the max number of numeric digits. The entered numeric data is left aligned. The TEXTBOX accepts only numeric values, and it ignores alpha keys. The operator must key in the decimal point if required.

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.

Select font from list: Arial (default) and Courier New in 10, 12, 14 and 16 point sizes.

The application can change the size of individual font using the SetFont function.

IND560 font options are Alpha, Chinese, Num1 and Num5. Please refer to **Appendix E**.

Color [Optional] Select text color.

Options: Black (default), Red, Blue, Green, White, Orange, Yellow, Purple.

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 tx0150. 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.

VCONSOLE

IND780 only

A Virtual Operator Console interfaces to the TaskExpert application through Shared Data. You can implement the virtual console using a TCP/IP or Serial Data connection that reads and writes Shared Data. The IND780 supports up to 3 virtual consoles. The data Shared Data blocks are am0100, am0200, and am0300 for virtual consoles 1, 2, and 3 respectively. Each block has Shared Data fields for LPRINT, operator input, and operator output messages.

A TaskExpert application uses the VCONSOLE command to assign a virtual console to itself.

Refer to the example provided in **Vconsole.zip**.

Properties:

Instance The instance number of the virtual console for this application. Legal instances are 0(none),1, 2, and 3.

Weight Display

Select options for the weight display. You must also define the rate either through Setup or though the DEFRATE command in order to see the Rate display.

Refer to the example provided in **WeightDisplay.zip**.

Properties:

Note: Properties denoted with an asterisk (*) are for IND780 only. The property denoted with two asterisks (**) is for IND560 only.

Display [Optional] Turns on or off the Weight Display.
Options: ON, OFF

Scale * [Optional] Selects all or just the active scale in the Weight Display.
Options: ALL, ACTIVE

Tare * [Optional] Selects the tare/rate display option.
Options: TARE, ALWAYS, NEVER, RATE

Compress * [Optional] Compresses or uncompresses the display.
Options: COMPRESS, UNCOMPRESS

Size ** [Optional] Selects the display size.
Options: SMALL, MEDIUM, LARGE

Size Scale * [Optional] Selects the size of the platform scale weight display.
Options: SMALL, MEDIUM, MEDIUMHALF, LARGE, LARGEHALF, LARGER, HUGE

Size Sum * [Optional] Selects the size of the sum weight display.
Options: SUM SMALL, SUM MEDIUM, SUM MEDIUMHALF, SUM LARGE, SUM LARGEHALF, SUM LARGER, SUM HUGE

Key Codes, IND780

The following table shows the mapping of the keypad keys for the IND780, including external keyboard keys that can simulate the keypad keys.

Keypad	External Keyboard Simulation Key/s	Key Code
A1	F10 Key / ALT and F1 key	SoftKey Table Or Inkey = 79
A2	F11 Key / ALT and F2 key	SoftKey Table Or Inkey = 7a
A3	F12 Key / ALT and F3 key	SoftKey Table Or Inkey = 7b
A4	APPS Key / ALT and F4 key	SoftKey Table Or Inkey = 5d

Keypad	External Keyboard Simulation Key/s	Key Code
SK1	F1 key	SoftKey Table Or Inkey = 70
SK2	F2 key	SoftKey Table Or Inkey = 71
SK3	F3 key	SoftKey Table Or Inkey =72
SK4	F4 key	SoftKey Table Or Inkey = 73
SK5	F5 key	SoftKey Table Or Inkey = 74
1	Numeric keypad 1 key	49
2	Numeric keypad 2 key	50
3	Numeric keypad 3 key	51
4	Numeric keypad 4 key	52
5	Numeric keypad 5 key	53
6	Numeric keypad 6 key	54
7	Numeric keypad 7 key	55
8	Numeric keypad 8 key	56
9	Numeric keypad 9 key	57
0	Numeric keypad 0 key	48
.	Decimal key	46
C (Clear)	Backspace key	8
Enter	Enter key	13
Left Arrow	Left arrow key	37
Right Arrow	Right arrow key	38
Up Arrow	Up arrow key	39
Down Arrow	Down arrow key	40
Scale Select	F6 key	1
Zero	F7 key	2
Tare	F8 key	3
Print	F9 key	4

Key Codes, IND560

The following table shows the mapping of the keypad keys for the IND560.

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
C (Clear)	8
Enter	13
Left Arrow	16
Right Arrow	18
Up Arrow	17
Down Arrow	19
Zero	30
Tare	24
Print	23

Chapter 5 Scale Commands

Note: Except as indicated, commands are common to IND560 **and** IND780.

Note: For the **Scale** property in commands common to IND560 and IND780, the only legal value for IND560 is 1. For the **Node** property in common commands, the only legal value for the IND560 is 0.

Command Reference

Notes: References to "specified" or "selected" scale refer only to IND780; these commands refer also to the IND560's single scale. Items marked with an asterisk (*) refer to IND780 only.

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 All Weight*	Get legal-for-trade weights for all scales on a node.
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.
Hi Filter*	Enable or disable the stability filter.
Select Scale*	Select a scale as the active scale on the Weight Display.
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.

Clear Tare

Clear tare – for scale (IND560) or for selected scale (IND780).

Refer to the example provided in **Clear Tare.zip**.

Properties:

Scale [Integer] Selects the scale. Legal values are: 1 to 5.

Node [Integer, Optional] Selects the node in the cluster. Legal values are: 0 & 1 to 20. 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

ConvertWt

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

Refer to the example provided in **Convert Weight.zip**.

Properties:

OldWt [Double] The weight value to be converted.

OldUnits The old weight units.

Options: None, pounds, kilograms, grams, metric tons, tons, troy ounces, penny weights, ounces

NewUnits The new weight units.

Options: None, pounds, kilograms, grams, metric tons, tons, troy ounces, penny weights, ounces

Scale [Integer, Optional] Selects the scale. Legal values are: 1 to 5. 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.

Define Rate

Define the rate calculation parameters for the specified scale.

Refer to the example provided in **Define Rate.zip**.

Properties:

Scale [Integer] Selects the scale. Legal values are: 1 to 5.

Node [Integer, Optional] Selects the node in the cluster. Legal values are: 0 & 1 to 20. 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.

Filter

Set filter parameters for scale. Restart filtering.

Refer to the example provided in **Filter.zip**.

Properties:

Scale [Integer] Selects the scale. Legal values are: 1 to 5.

Node [Integer, Optional] Selects the node in the cluster. Legal values are: 0 & 1 to 20. 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 (IND560), 2, 4, 6, 8 and 10 (IND780). 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 1 to 183. 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 NOT 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

Get All Weight

IND780 only

Get legal-for-trade weights for all scales on a node.

Refer to the example provided in **Get All Weight.zip**.

Properties:

Node [Integer] Selects the node in the cluster. Legal values are 0 & 1 to 20. 0 refers to the local node.

Netweight1 [String or double variable, Output/Result variable] Net weight of the scale 1. 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.

Tareweight1 [String or double variable, Output/Result variable] Tare weight of scale 1. 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.

Netweight(2-5) [String or double variable only, Output/Result variable, Optional] Net weights for the next available scales. The data type depends on the data type of the variable.

Tareweight(2-5) [String or double variable only, Output/Result variable, Optional] Tare weights for the next available scales. The data type depends on the data type of the variable.

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.

Refer to the example provided in **Get Average Weight.zip**.

Properties:

Scale [Integer] Selects the scale. Legal values are: 1 to 5.

Node [Integer] Selects the node in the cluster. Legal values are 0 & 1 to 20. 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.

Get Rate

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

Refer to the example provided in **Get Rate.zip**.

Properties:

Scale [Integer] Selects the scale. Legal values are 1 to 5.

Node [Integer, Optional] Selects the node in the cluster. Legal values are 0 & 1 to 20. 0 or non-existent parameter refers to the local node.

Rate [String, Output/Result variable] Returns the rate in DOUBLE floating point format.

Get Rate$

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

Refer to the example provided in **Get Rate.zip**.

Properties:

Scale [Integer] Selects the scale. Legal values are 1 to 5.

Node [Integer, Optional] Selects the node in the cluster. Legal values are 0 & 1 to 20. 0 or non-existent parameter refers to the local node.

Rate [Double variable, Output/Result variable] Returns the rate in STRING format.

Get Setpoint

Get the feed status of a setpoint.

Refer to the example provided in **Get Setpoint Status.zip**.

Properties:

Note: In the IND560, 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 (IND560), 1 to 17 (IND780).

Node [Integer, Optional] Selects the node in the cluster. Legal values are 0 & 1 to 20. 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:
*** IND780 only**

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
*Weigh-In Feeding	Bit6 0 = Weigh-Out Feeding 1 = Weigh-In Feeding
*Dump to Empty Feeding	Bit7 0 = no, 1 = In Progress

Byte 1

*Dump to Empty Draining	Bit8 0 = no, 1 = In Progress
Pause	Bit9 1 = Pause State
*In Progress	Bit10 1 = Feed In Progress. This bit is an "OR" of all active Target bits.

Note: When using this function with the IND560, the Tolerance bits are only returned when the IND560 Target Mode is set to Over/Under.

Get Weight

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

Refer to the example provided in **Get Weight.zip**.

Properties:

Scale [Integer] Selects the scale. Legal values are 1 to 5.

Node [Integer] Selects the node in the cluster. Legal values are 0 & 1 to 20. 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 = ozt (troy ounces)
2 = kg (kilograms)	7 = dwt (penny weight)
3 = g (grams)	8 = oz (ounces)
4 = t (metric tons)	

Hi Filter

IND780 only

Enable or disable the stability filter. Restart filtering. Ultra-Stability Filtering algorithm is for use in transaction applications where it is very difficult to achieve stable weight readings due to excessive motion on the scales. Examples are truck scales in very windy locations and livestock weighing scales. The Ultra-Stability filtering algorithm uses the standard low-pass filtering as long as there is a rapid motion on the scale so that the operator can also observe the weight changing. When the motion begins to diminish, this algorithm switches to a very stiff filter that strongly dampens any noise on the scale. Then, the operator can record a stable weight reading. Process weighing applications cannot use the ultra-stability filter, since the non-linear action of the filter switching may cause inaccurate cutoffs in batching or filling.

Refer to the example provided in **Hi FIlter.zip**.

Properties:

Scale [Integer] Selects the scale. Legal values are 1 to 5.

Node [Integer, Optional] Selects the node in the cluster. Legal values are 0 & 1 to 20. 0 or a non-existent parameter refers to the local node.

Ultra Stability Enables or disables the stability filter. Options: Enable, Disable.

Status [Integer, Output/Result variable, Optional] The output status for the operation. Please refer to the status definition under the Filter function block.

Select Scale

IND780 only

Select a scale as the active scale on the Weight Display

Refer to the example provided in **Select Scale.zip**.

Properties:

Scale [Integer] Selects the scale. Legal values are 1 to 5.

Node [Integer, Optional] Selects the node in the cluster. Legal values are 0 & 1 to 20. 0 or a non-existent parameter refers to the local node.

Status [Integer, Output/Result variable, Optional] The output status for the operation. Please refer to the status definition under the Set Units function block.

Refer to the example provided in **SelectedScale.zip**.

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 92 comparisons per second. It shuts off the feed when system reaches the target conditions defined in the setpoint data.

Refer to the example provided in **Setup Setpoint.zip**.

Properties:

Scale [Integer] Selects the scale. Legal values are 1 to 5.

Node [Integer, Optional] Selects the node in the cluster. Legal values are 0 & 1 to 20. 0 or a non-existent parameter refers to the local node.

Note: By default, Setpoints 1 to 5 are assigned to Scales 1-5. If another setpoint is assigned to a scale (for example, Setpoint 8 is assigned to Scale 1), the scale must be restarted. This can be done by triggering the Apply Setup SD variable wc--08 for the affected scale.

Setpoint [Integer] Number of the setpoint that this function initializes. Legal values are: 1 (IND560), 1 to 17 (IND780).

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.

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 Defines how the terminal displays the setpoint operation on the SmartTrac display.

Options: Turn-off, Bar Graph, Cross Hairs, 3-zone, Default Setting

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

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.

Refer to the example provided in **Setpoint Target.zip**.

Properties:

Setpoint [Integer] The number of the setpoint that this function initializes. Legal values are: 1 (IND560), 1 to 17 (IND780).

Node [Integer, Optional] Selects the node in the cluster. Legal values are 0 & 1 to 20. 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.

Set Units

Select the primary or secondary units operation for a scale.

Refer to the example provided in **Set Units.zip**.

Properties:

Scale [Integer] Selects the scale. Legal values are 1 to 5.

Node [Integer, Optional] Selects the node in the cluster. Legal values are 0 & 1 to 20. 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

Stop Setpoint

Stop the current feed for this setpoint.

Refer to the example provided in **Stop Setpoint.zip**.

Properties:

Setpoint [Integer] The number of the setpoint that this function initializes. Legal values are 1 to 17.

Node [Integer, Optional] Selects the node in the cluster. Legal values are 0 & 1 to 20. 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.

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.

Refer to the example provided in **Switch Resolution.zip**.

Properties:

Scale [Integer] Selects the scale. Legal values are 1 to 5.

Node [Integer] Selects the node in the cluster. Legal values are 0 & 1 to 20. 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.

Tare

Tare scale (IND560) or selected scale (IND780). If the function call contains a tare value, it will be used. Otherwise, perform a pushbutton tare.

Refer to the example provided in **Tare.zip**.

Properties:

Scale [Integer] Selects the scale. Legal values are: 1 to 5.

Node [Integer, Optional] Selects the node in the cluster. Legal values are: 0 & 1 to 20. 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.

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.

Refer to the example provided in **Weight String.zip**.

Properties:

Weight [Float] The floating-point weight value to convert.

Scale [Integer] Selects the scale. Legal values are 1 to 5. 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, troy ounces, penny weights, ounces. 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.

Zero Scale

Attempt to Zero the specified scale.

Refer to the example provided in **Zero.zip**.

Properties:

Scale [Integer] Selects the scale. Legal values are 1 to 5.

Node [Integer, Optional] Selects the node in the cluster. Legal values are 0 & 1 to 20. 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

Chapter 6 Interface Commands

Note: Except as indicated, commands are common to IND560 and IND780.

Note: All PLC commands apply to IND780 only

Command Reference

File Commands

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

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.

PLC Commands

Command	Usage
Get Cyclic PLC Data	Gets the cyclic input data sent from the host PLC to the terminal.
Get PLC Float Data	Formats and retrieves floating-point data.
Get PLC Int Data	Formats and retrieves integer data.
Get PLC String Data	Formats and retrieves string data.
PLCCONFIG	Sets configuration data for interfacing to the PLC.

Command	Usage
Set Cyclic PLC Data	Sends cyclic output data from the terminal to the host PLC.
Set PLC Float Data	Formats and inserts floating-point data into the CyclicToPLC%[] variable array.
Set PLC Int Data	Formats and inserts integer data into the CyclicToPLC%[] variable array.
Set PLC String Data	Formats and inserts string data into the CyclicToPLC%[] variable array.

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.

Email Command

Command	Usage
Email	Send an email message to a single recipient.

FTP Client Commands

Command	Usage
FTPOpen	Initiates a client connection to a remote FTP server.
FTPClose	Terminates a client connection with a remote FTP server.
FTPSend	Sends a local file to the remote terminal.
FTPRecv	Receives a file from the remote terminal.

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.

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.

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.

Connection Commands

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

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.

Refer to the example provided in **SocketProg.zip**.

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.

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.

Refer to the example provided in **File Creation.zip**.

Properties:

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

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

Close Socket

The Close Socket command allows the TaskExpert application to close an established TCP/IP socket connection.

Refer to the example provided in **ClientProg.zip**.

Properties:

Socket [Integer, Optional] Socket number to close. If a socket not specified, the function closes all sockets; else if socket = 0, the function closes the listen server connection; else the function closes specified connection socket.

Status [Integer variable, Output/Result variable] Status of the socket. Please refer to TCP FUNCTION STATUS (LISTEN)

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.

Refer to the example provided in **ClientProg.zip**.

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).

Custom Print

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

Refer to the example provided in **CustomPrint.zip**.

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 & 1 to 20. 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.

Define Input Connection

Define an input connection in the Data Connections table in Shared Data. If there is already a duplicate entry, do not insert the new entry but return a successful status anyway. Otherwise, insert the new entry in the first unused Data Connection table entry. This command only writes to the local Shared Data.

Refer to the example provided in **DefInConn.zip**.

Properties:

Input Type [Input] Type of input connection.
Options: None, Scale Commands (CTPZ-style), Scale Commands (SICS Slave Level 0 & 1), Bar Codes, [Remote Keyboard (**IND780 only**)].

Template [Integer] Applicable only to bar code connections.
Options: None, Programmable bar code template1, Programmable bar code template2.

Port [Integer] IO port number. There is only one IO port per input data connection.
1-3 are Serial Ports 1-3 (**IND560**)
1-6 are Serial Ports 1-6 (**IND780**)
7-12 are USB Ports 1-6 (**IND780**)

Status [Integer variable, Output/Result variable, Optional] Return status of this function block. Please refer to the PRINTING STATUS (DEMAND) statuses.

Define Output Connection

Define an output connection in the Data Connections table in Shared Data. If there is already a duplicate entry, don't insert the new entry but return a successful status anyway. Otherwise, insert the new entry in the first unused Data Connection table entry. This command only writes to the local Shared Data.

Refer to the example provided in **DefOutConn.zip**.

Properties:

Output Type [Integer] The type of output connection.

Options (**IND560**): None, Scale transaction and custom demand print, Continuous output, Continuous demand print, Reports, Action log output, Short continuous, Totals reports

Options (**IND780**): None, Demand Print, Continuous Print, Multi-continuous Print, Multi-continuous Print 1

Trigger [Integer] Device or function that triggers the output:

Options (**IND560**): None, Scale 1, Custom Print 1-3

Options (**IND780**): None, Scales 1-5, Custom Print 1-10, Flow Meter 1-12

Template [Integer] Print Template associated with the Data Connection.

Options (**IND560**): Standard template, Programmable templates 1-5

Options (**IND780**): Standard template, Programmable Templates 1-10

Port1 [Integer] IO port for the output connection.

Port2 [Integer, Optional] IO port for the output connection.

Port3 [Integer, Optional] IO port for the output connection.

(**IND560**) 1-3 are Serial Ports 1-3

14-15 are TCP/IP Demand Print Message streams 2-3 for remote data connections. An IND560 client or PC client application must connect to the Shared Data Server to receive data from this the output connection.

(**IND780**) 1-6 are Serial Ports 1-6

7-12 are USB Ports 1-6

13-15 are TCP/IP Demand Print Message streams 1-3 for remote data connections. An IND780 client or PC client application must connect to the Shared Data Server to receive data from this the output connection.

(**IND560 and IND780**)16 is TCPIP message stream for continuous output

(**IND560 and IND780**) 17 is the TCP/IP message stream for an EPRINT connection to serve as a second Shared Data Server port. This connection supports the legacy JagXTREME Console Print server connection for continuous output and demand print.

Status [Integer variable, Output/Result variable, Optional]

Return status of this function block. Please refer to the PRINTING STATUS (DEMAND) statuses

Demand Print

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

Refer to the example provided in **Demand Print.zip**.

Properties:

Scale [Integer] Selects the scale. Legal values are 1 to 5.

Node [Integer] Selects the node in the cluster. Legal values are 0 & 1 to 20. 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

Flush

Interfaces → Flush

Discards received data in the BIOS serial input buffer.

Refer to the example provided in **File Creation.zip**.

Properties:

Serial Port Opened serial port number in "#<Device Number>" format

Email

Miscellaneous → Email

Send an email message to a single recipient.

Note: For calling parameters To, From and Subject, the maximum length is 39 characters plus a terminating null character.

Refer to the example provided in **Email.zip**.

Properties:

To [String] The email recipient.

From [String] The email address of the sender

Subject [String] The subject of the email message

Body [String] The email message text

Attachment [String] **IND780 only**: The name and path of a file to attach to the email

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

Return values:

SUCCESS	0
Email disabled or other error	-1

FTP Client Functions

FTP Client functions allow an application, running in its own thread, to connect to and transfer files to and from a remote machine.

The FTP Client functions do not support concurrent "event-driven" operations in their own TaskExpert thread while an FTP Client function is running; but the "event-driven" operations do run after the FTP Client function is complete. Since the FTP Client functions can take some time complete, the TaskExpert thread delays running the event-driven operation until the FTP Client operation completes.

You may run an FTP file transfer concurrently with your main application by starting a separate TaskExpert thread for the FTP Client functions using the TaskExpert Multi-Tasking statements. In this case, the FTP Client functions do not delay event-driven operations from running in main application thread.

Note: In IND780 Firmware versions 5.1.xx, the FTP Client Commands only function with Windows-based FTP servers.

Refer to the example provided in **FTP.zip**.

FTPOpen

The FTPOpen function initiates a client connection to a remote FTP server.

Properties:

IP address [String] Host IP address

Username [String] Host FTP server username

Password [String] Host FTP server password

Status [Integer variable, Output/Result variable, Optional] The return value is an integer variable that is the status of the connection.

Return Status:

```
Success                                    0
Attempted multiple connections,
   only one connection permitted          -1
Invalid IP Address format                 -2
Timeout occurred during a receive        -11
Socket error occurred during a receive   -12
Command socket closed during a receive   -13
```

Also see **Winsock Error Codes** for other possible return values.

FTPClose

The FTPClose function terminates a client connection with a remote FTP server.

Properties:

Status [Integer variable, Output/Result variable, Optional] The return value is an integer variable that is the status of the function block.

Return Status:

Success	0
Quit command failed	-10

Also see **Winsock Error Codes** for other possible return values.

FTPSend

The FTPSend function sends a local file to the remote machine.

Properties:

SourcePath [String] Source path and filename.

DestinationPath [String] Destination path and filename.

Status [Integer variable, Output/Result variable, Optional] The return value is an integer variable that is the status of the function block.

Return Status:

Success	0
Source or destination file path not found	-3
No connection established	-4
Cannot change working directory	-8
Data socket connection failed	-9
Timeout occurred during a receive	-11
Socket error occurred during a receive	-12
Command sockect closed during a receive	-13
No data termination response from server	-14

Also see **Winsock Error Codes** for other possible return values.

Note: The IND780 automatically adds "\storage card" to the beginning of the source path and file name. For example, if a file named *batch1.dat* is located in the main directory of the storage card, all that would need to be entered for the Source Path parameter would be "\batch1.dat". When the IND780 runs this function, the actual directory path and file name it would use to locate the file would be "\storage card\batch1.dat".

If the file is to be renamed when it is sent to the remote FTP server, the new file name can be included in the Destination Path. For example, if *batch1.dat* is to be transferred and renamed *IND780batch.dat*, "\IND780batch.dat" should be entered as the DestinationPath.

FTPRecv

The FTPRecv function receives a file from the remote machine.

Note: The IND780 automatically adds "\storage card" to the beginning of the receive path and filename. For example, if a file named *newBatch.dat* located on the remote FTP server is to be placed in the main directory of the Storage Card using the same name, the Local Receive Path parameter would be simply "\". In this case, the actual directory path and filename the IND780 would use to store the file in memory would be "\storage card\". The file *newBatch.dat* would then be stored on the Storage Card.

If the file is to be renamed when it is saved to the IND780 Storage Card, the new filename should be included in the Local Receive Path parameter. For example, to rename the file *batch1.dat*, "\batch1.dat" should be entered as the Local Receive Path parameter.

Properties:

LocalReceivePath [String] New filename, if required, and path where the file will be stored locally.

RemoteSourcePath [String] Remote source path and filename.

Status [Integer variable, Output/Result variable, Optional] The return value is an integer variable that is the status of the function block.

Return Status:

```
Success                                  0
Source or destination file path not
   found                                -3
No connection established               -4
File write is not permitted             -5
File is read only                       -6
File receive error                      -7
Cannot change working directory         -8
Data socket connection failed           -9
Timeout occurred during a receive      -11
Socket error occurred during
   a receive                           -12
Command sockect closed during
   a receive                           -13
No data termination response
   from server                         -14
```

Also see **Winsock Error Codes** for other possible return values.

Winsock Error Codes

Value	Description
10004	Interrupted function call.
10013	Permission denied. An attempt was made to access a socket in a way forbidden by its access permissions.
10014	Bad address. The system detected an invalid pointer address in attempting to use a pointer argument of a call. This error occurs if an application passes an invalid pointer value, or if the length of the buffer is too small.
10022	Invalid argument
10024	Too many open files. Too many open sockets.
10035	Resource temporarily unavailable. This error is returned from operations on non-blocking sockets that cannot be completed immediately.
10036	Operation now in progress.
10037	Operation already in progress.

Value	Description
10038	Socket operation on non-socket.
10039	Destination address required.
10040	Message too long.
10041	Protocol wrong type for socket.
10042	Bad protocol option.
10043	Protocol not supported.
10044	Socket type not supported.
10048	Address already in use. Typically, only one usage of each socket address (protocol/IP address/port) is permitted.
10049	Cannot assign requested address.
10050	Network is down.
10051	Network is unreachable
10052	Network dropped connection on reset.
10053	Software caused connection abort. An established connection was aborted by the software in your host machine, possibly due to a data transmission time-out or protocol error.
10054	Connection reset by peer. An existing connection was forcibly closed by the remote host.
10055	No buffer space available. An operation on a socket could not be performed because the system lacked sufficient buffer space or because a queue was full.
10056	Socket is already connected. A connect request was made on an already-connected socket.
10057	Socket is not connected. A request to send or receive data was disallowed because the socket is not connected and no address was supplied.
10058	Cannot send after socket shutdown. A request to send or receive data was disallowed because the socket had already been shut down in that direction with a previous shutdown call.
10060	Connection timed out. A connection attempt failed because the connected party did not properly respond after a period of time, or the established connection failed because the connected host has failed to respond.
10061	Connection refused. No connection could be made because the target machine actively refused it.
10064	Host is down. A socket operation failed because the destination host is down.

Value	Description
10065	No route to host. A socket operation was attempted to an unreachable host.
10067	Too many processes. A Windows Sockets implementation may have a limit on the number of applications that can use it simultaneously.
10091	Network subsystem is unavailable. This error is returned if the Windows Sockets implementation cannot function at this time because the underlying system it uses to provide network services is currently unavailable.
10092	Winsock.dll version out of range. The current Windows Sockets implementation does not support the Windows Sockets specification version requested by the application.
10101	Graceful shutdown in progress.
10109	Class type not found. The specified class was not found.
11001	Host not found. No such host is known.

Get Cyclic PLC Data

Get the cyclic input data sent from the host PLC to the IND780. The PLC sends cyclic data to the IND780 on a regular scheduled basis, such as, once every 50 milliseconds. Most times, however, the data is the same as on the previous transmission. The TaskExpert application can setup an event so that it receives a callback whenever the IND780 receives new data.

Refer to the example provided in **GetCyclic.zip**.

Properties:

Status [Integer variable, Output/Result variable] Status contains the length of the input data. The GetCyclic function returns the PLC output data in the variable integer array cyclicFromPLC[], that PLCCONFIG automatically creates. GetCyclic copies the character data from the PLC into the integer array. If cyclicFromPLC[] does not exist, the GetCyclic function automatically creates the variable with its maximum length of 500. If the application creates the variable, it can create the variable to the length of the PLC cyclic input. GetCyclic copies the input character data from the PLC into the integer array.

If Status > 0, status contains the length of the input data.

If Status < 0, status contains an error status:

```
PLC_FUNCTION_STATUS
PLC_FUNCTION_COMPLETED_SUCCESSFULLY     0
PLC_OFFLINE                            -1
PLC_NO_NEW_INPUT_DATA                  -2
PLC_CYCLIC_OUTPUT_NOT_DEFINED          -3
PLC_EXPLICIT_OUTPUT_NOT_DEFINED        -4
PLC_OUTPUT_LENGTH_TOO_LONG             -5
```

Get IP

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

Refer to the example provided in **ClientProg.zip**.

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.

Get PLC Float Data

Format and retrieve floating-point data, after the GetCyclic()has received the PLC output buffer into the CyclicFromPLC variable.

Refer to the example provided in **GetPLCFloat.zip**.

Properties:

Offset [Integer] Specifies the location of the floating-point data in the PLC output buffer.

Float Data [Float variable, Output/Result variable]
Returns the data in a floating-point variable.

Get PLC Int Data

Format and retrieve integer data, after the GetCyclic()has received the PLC output buffer into the cyclicFromPLC%[] variable.

Refer to the example provided in **GetPLCInt.zip**.

Properties:

Offset [Integer] Specifies the location of the integer data in the PLC output buffer.

Int Data [Integer variable, Output/Result variable]
Returns the data in an integer variable.

Get PLC String Data

Format and retrieve string data, after the GetCyclic()has received the PLC output buffer into the cyclicFromPLC%[] variable.

Refer to the example provided in **GetPLCString.zip**.

Properties:

Offset [Integer] Specifies the location of the string data in the PLC output buffer.

Length [Integer] Specifies length of the string data.

String Data [String variable, Output/Result variable]
Returns the data in a string variable.

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.

Refer to the example provided in **FileCreation.zip**.

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.

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.

Refer to the example provided in **SocketProg.zip**.

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
```

Open COM

Prepares a serial port for use as a file device. You can access a serial port that you have 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 setup as a continuous output connection or as an input connection. If the serial port is on the local terminal, you can access the serial port even if it is not set up in a connection.

The OPEN COM command allows you to specify the remote terminal address and the serial port address on the remote terminal. When it issues the OPEN COM command, the TaskExpert program is establishing exclusive access to the remote serial port as long as it has the serial port open. If another terminal has already opened the serial port, the TaskExpert program will get an error status back indicating there is a file-sharing error. In order to effectively share a serial port among several terminals, you should open the serial port, quickly perform the I/O, and then close the serial port to make it available to another terminal.

Refer to the example provided in **File Creation.zip**.

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 7.
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 statements transmit a NULL character embedded in each print statement.
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.
Remote Terminal	[String constant only, Optional] Terminal addresses which specify remote terminal containing the remote serial port.

PLCCONFIG

Set configuration data for interfacing to the PLC. The TaskExpert program must execute the PLCCONFIG function first before attempting any other PLC communication commands. It must re-execute the command each time the program re-starts. The data lengths specified in the PLCCONFIG function must match exactly with the lengths configured in the PLC lengths in order for the communications to be successful.

Refer to the example provided in **PLCConfig.zip**.

PLCCONFIG automatically creates two fixed-name variables that the TaskExpert uses for setting and retrieving data from a PLC. They are integer array variables with a maximum length of 500.

- cyclicFromPLC%[]
- cyclicToPLC%[]

Properties:

Into PLC Length [Integer] The length of the data that the terminal sends to the PLC.

Out From PLC Length [Integer] The length of the data that the PLC sends to the terminal.

Big Endian [Integer] Indicates that the PLC uses the Big Endian data format. It is an optional parameter. When the Big Endian is set, the TaskExpert PLC functions automatically switch the byte-ordering of integer and floating point data used in communications with the PLC.

Status [Integer variable, Output/Result variable] Status of PLC Configuration.

Return value:

PLC_FUNCTION_STATUS	
PLC_FUNCTION_COMPLETED_SUCCESSFULLY	0
PLC_OFFLINE	-1
PLC_NO_NEW_INPUT_DATA	-2
PLC_CYCLIC_OUTPUT_NOT_DEFINED	-3
PLC_EXPLICIT_OUTPUT_NOT_DEFINED	-4
PLC_OUTPUT_LENGTH_TOO_LONG	-5

Print

Interfaces > Print

PRINT outputs the data to the specified serial port.

Refer to the example provided in **PrintUsing.zip**.

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.

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.

Refer to the example provided in **DIO.zip**.

Properties:

Node [Integer, Optional] Selects the node in the cluster. Legal values are 0 & 1 to 20. 0 or non-existent parameter refers to the local node.

Slot [Integer] The local board slot or remote IO slot that contains the Discrete IO.
(IND560) There is 1 legal local board slot: 1, and 3 legal remote IO slots: 2 – 4
(**IND780**) There are 2 legal local board slots: 5 – 6, and 8 legal remote IO slots: 7 – 14.

Discrete Output [Integer] The discrete output's position within the slot.
(**IND560**) Legal positions for the local board slot are 1 – 6. Legal positions for the remote IO are: 1 – 6.
(**IND780**) Legal positions for the local board slot are: 1 – 4. 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.

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.

Refer to the example provided in **DIO.zip**.

Properties:

Node [Integer, Optional] Selects the node in the cluster. Legal values are 0 & 1 to 20. 0 or non-existent parameter refers to the local node.

Slot [Integer] Local board slot or remote IO slot that contains the Discrete IO.
(**IND560**) There is 1 legal board local slot: 1. There are 3 legal remote IO slots: 2 – 4.
(**IND780**) There are 6 legal board local slots: 1 – 6. There are 8 legal remote IO slots: 7 – 14.

Discrete Input [Integer] Discrete input's position within the slot.

(**IND560**) Legal positions for the local board slot are: 1 – 4. Legal input positions for a remote IO slot are: 1 – 4.
(**IND780**) Legal positions for the local board slot are: 5 – 6. Legal input positions for a remote IO slot are: 1 – 4.

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

Receive

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

Refer to the example provided in **SocketProg.zip**.

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 (200 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.

RecvArray

IND780 Only

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

Refer to the example provided in **RecvArray.zip**.

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.

Send

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

Refer to the example provided in **ClientProg.zip**.

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).

SendArray

IND780 Only

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.

Refer to the example provided in **RecvArray.zip**.

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.

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.

Refer to the example provided in **AnalogOutput.zip**.

Properties:

Channel [Integer] Analog Output channel number. Legal values are 1 to 4.

Value [String] Specifies Preset Zero value.

Status:

Success	0
ERR_INVALID_CHANNEL	-1

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.

Refer to the example provided in **AnalogOutput.zip**.

Properties:

Channel [Integer] Analog Output channel number. Legal values are 1 to 4.

Value [String] Specifies Preset Span value.

Status:

Success	0
ERR_INVALID_CHANNEL	-1

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.

Refer to the example provided in **AnalogOutput.zip**.

Properties:

Channel [Integer] Analog Output channel number. Legal values are 1 to 4.

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

Set Cyclic PLC Data

Send cyclic output data from the IND780 to the host PLC. The IND780 sends cyclic data to the host PLC on a regular scheduled basis, such as, once every 50 milliseconds. The cyclic data that the IND780 sends typically contains weight data. The PLC subsystem sends the latest cyclic data that the application has set with the SetCyclic command on each scheduled message that it sends to the host.

CyclicToPLC%[] is an integer array of maximum length 500. It contains the cyclic data to send to the PLC. The SetCyclic function copies the integer array into a character buffer to send to the PLC. cyclicToPLC%[] is a fixed-name variable that the application does not name in the calling parameters.

Refer to the example provided in **SetCyclic.zip**.

Properties:

Status [Integer variable, Output/Result variable, Optional]

Status of this function call. Please refer to the PLC FUNCTION STATUS (PLCCONFIG).

Set Discrete IO Option

Interfaces — Set Discrete IO Option

Turn a discrete output on or off.

Refer to the example provided in **DIO.zip**.

Properties:

Node [Integer] Selects the node in the cluster. Legal values are 0 & 1 to 20. 0 or non-existent parameter refers to the local node.

Slot [Integer] Local board slot or remote IO slot that contains the Discrete IO.
(**IND560**) There is 1 legal board local slot: 1. There are 3 legal remote IO slots: 2 – 4.
(**IND780**) There are 2 legal board local slots: 5 – 6. There are 8 legal remote IO slots: 7 – 14.

Discrete Output [Integer] Discrete output's position within the slot.
(**IND560**) Legal positions for the local board slot are: 1-6. Legal positions for a remote IO slot are: 1 – 6.
(**IND780**) Legal positions for the local board slot are: 1 – 4. Legal positions for a remote IO slot are: 1 – 6.

ON/OFF [Integer] Indicates whether to turn it on or off.
Options: ON, OFF.

Status [Integer variable, Output/Result variable]
Return value of this function block. Please refer to BASIC FUNCTION STATUS.

Set PLC Float Data

Interfaces — Set PLC Float Data

Format and insert floating-point data into the cyclicToPLC%[] variable array, before the SetCyclic() copies this variable array data into PLC output buffer and transmits it to the PLC.

Refer to the example provided in **SetPLCFLoat.zip**.

Properties:

Float Data [Float] Specifies the floating-point data that gets inserted into the PLC input buffer.

Offset [Integer] Specifies the location of the floating-point data in the PLC input buffer.

Status [Integer variable, Output/Result variable, Optional]
Status of this function call. Please refer to the PLC FUNCTION STATUS (PLCCONFIG).

Set PLC Int Data

Format and insert integer data into the cyclicToPLC%[] variable array, before the SetCyclic() copies the variable array data into PLC output buffer and transmits it to the PLC.

Refer to the example provided in **SetPLCInteger.zip**.

Properties:

Int Data [Integer] Specifies the integer data that gets inserted into the PLC input buffer.

Offset [Integer] Specifies the location of the integer data in the PLC input buffer.

Status [Integer variable, Output/Result variable, Optional]
Status of this function call. Please refer to the PLC FUNCTION STATUS (PLCCONFIG).

Set PLC String Data

Format and insert string data into the cyclicToPLC%[] variable array, before the SetCyclic() copies this variable array data into PLC output buffer and transmits it to the PLC.

Refer to the example provided in **SetPLCString.zip**.

Properties:

String Data [Integer] Specifies the string data that gets inserted into the PLC input buffer.

Offset [Integer] Specifies the location of the integer data in the PLC input buffer.

Length [Integer] Optionally specifies maximum length of the string.

Status [Integer variable, Output/Result variable, Optional]
Status of this function call. Please refer to the PLC FUNCTION STATUS (PLCCONFIG).

Set Template

Set a Print Template in Shared Data. A Print Template can be up to 1000 bytes long. Since the maximum BASIC string size is 200 bytes, you may need 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.

Refer to the example provided in **SetTemplate.zip**.

Properties:

Template [Integer] Selects the template number.
(**IND560**) Legal values are 1 to 5.
(**IND780**) Legal values are 1 to 10.

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.

Socket

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

Refer to the example provided in **ClientProg.zip**.

Properties:

Socket Type [Optional, Integer] Type of socket to create.
0 – TCP socket
1 – UDP socket
Note: If left blank, defaults to TCP.

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).

Width

Assigns an output line width to the LPRINT device, serial port, or a file. Used to limit the line lengths in a file containing a report. Line lengths beyond the established width are wrapped to the next line. The default width is 80 characters.

Refer to the example provided in **Width.zip**.

Properties:

File Number [Optional] The number of an open file. If File Number is not specified, WIDTH applies to the LPRINT device.

Width [Integer] The desired width in columns.

Width In

Allows the user to dynamically reassign the maximum serial input length, as it is defined in the OPEN COM.

Refer to the example provided in **WidthIn.zip**.

Properties:

File Number Open serial I/O device.

Length [Integer] The desired length. Legal values are 0 to 80.

Chapter 7 File Commands

Note: Except as indicated, commands are common to IND560 and IND780.

Command Reference

Command		Usage
IND560	**IND780**	
Close		Closes an open file or serial port.
Data		Specifies values to be read by READ statements.
	DelRec	Deletes a record from the indexed sequential file.
	Field	Defines the structure of records to be used in indexed-sequential and random-access file buffers.
	File Export	Create a real copy of an IND780 phantom system file and place it in the specified path and filename.
	File Import	Take data from an existing user file and use it to update the IND780 Shared Data or the Standard Database using phantom file names.
	FileCopy	Copies a file locally.
	Get	Reads a record from a random access file by record number into fields defined by the field statement.
	Indexed	Identifies which field in the record is the index key.
Input		Reads input from the keyboard, serial port, or a sequential file.
	Kill	Deletes either a file or an entire file path
	LSET	Moves the value of an expression or variable into a field in a random-access file buffer in preparation for a PUT statement. (Left-Justified)
Open File		Accesses a file.
Print		Outputs data to the specified serial port or writes data to a sequential file.
	Put	Writes records to a random access file.
Read		Reads values from a DATA statement and assigns them to variables.
	RSET	Moves the value of an expression or variable into a field in a random-access file buffer in preparation for a PUT statement. (Right-Justified)
Restore		Allows DATA statements to be reread from a specified line.
	SortRec	Sorts the file records in sequential order by key.
Swap		Exchanges the values of two variables.
Write		Outputs delimited data to the sequential file.

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.

Refer to the example provided in **File Creation.zip**.

Properties:

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

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

Data

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

Refer to the example provided in **DataRead.zip**.

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(" ")

DelRec

IND780 only

Deletes a record from the indexed sequential file. The program must set the logical index into the key field or FIELD variables. The DELREC function searches the file for a record containing the logical key. If it finds the record, the DELREC function deletes the record in the FIELD variables. Otherwise, it generates a "RECORD NOT FOUND" error.

Refer to the example provided in **FileControls.zip**.

Properties:

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

Field

IND780 only

Defines the structure of records to be used in indexed-sequential and random-access file buffers. Records contain various fields. Each field is a location in a record that can be accessed by a field name.

Refer to the example provided in **IndexedFile.zip**.

Properties:

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

Field Field width and variable that identifies the field and contains field data.

Example:

```
"30 AS Name$,50 AS address$"
```

File Export

IND780 only

Create a real copy of an IND780 phantom system file and place it in the specified path and filename. The phantom file names represent system data that resides in the IND780 Shared Data or in the IND780 Standard Database. The FTP Server provides a similar capability of copying the phantom files to a real file remotely.

Properties:

Phantom Filename The path and name of the phantom system file. Possible values are:

- \Storage Card\Terminal\HIS\Alibi.csv
- \Storage Card\Terminal\HIS\Change.csv
- \Storage Card\Terminal\HIS\Error.csv
- \Storage Card\Terminal\HIS\Maintenance.csv
- \Storage Card\Terminal\SD\BRAM.dmt
- \Storage Card\Terminal\SD\EEPROM.dmt
- \Storage Card\Terminal\SD\FLASH.dmt
- \Storage Card\Terminal\TABLES\Standard_Ax.csv (replace x with table number 0 through 9.)

Return Value

SUCCESS	0
ERR_CANNOT_ACCESS_FILE	-2
ERR_DISPLAY_UNICODE_CONVERSION	-7
ERR_PATH_NOT_FOUND	-13
ERR_ACCESS_DENIED	-14

Example:

```
phantom$ = "\Storage Card\Terminal\SD\BRAM.dmt"
new$ = "\Storage Card\TaskExpert\Data\myBRAM.dmt"
```

File Import

IND780 only

Take data from an existing user file and use it to update the IND780 Shared Data or the Standard Database using phantom file names. The phantom file names represent the system data that resides in the IND780. The FTP Server provides a similar capability to update the system data represented by the phantom file names remotely.

Properties:

Exist Filename The path and name of the file from which data is to be retrieved.

Phantom Filename The path and name of the phantom system file. Possible values are:

\Storage Card\Terminal\SD\BRAM.dmt
\Storage Card\Terminal\SD\EEPROM.dmt
\Storage Card\Terminal\SD\FLASH.dmt
\Storage Card\Terminal\TABLES\Standard_Ax.csv (replace x with table number 0 through 9.)

Return Value

SUCCESS	0
ERR_CANNOT_ACCESS_FILE	-2
ERR_DISPLAY_UNICODE_CONVERSION	-7
ERR_PATH_NOT_FOUND	-13
ERR_ACCESS_DENIED	-14

Example:

```
exist$ = "\Storage Card\TaskExpert\Data\myBRAM.dmt"
phantom$ = "\Storage Card\Terminal\SD\BRAM.dmt"
```

FileCopy

IND780 only

Copy a file locally (compact flash or USB memory).

Refer to the example provided in **FileCopy.zip**.

Properties:

ExistFilename [String] The path and name of the file to be copied

NewFilename [String] The path and name of the new file. If there is already a file with the name specified in NewFilename, it will be overwritten if it is not Read Only.

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
ERR_CANNOT_ACCESS_FILE	-2
ERR_DISPLAY_UNICODE_CONVERSION	-7
ERR_PATH_NOT_FOUND	-13
ERR_ACCESS_DENIED	-14

Example 1, copying to/from the Compact Flash card:

```
"\storage card\myFileName.txt"
```

Example 2, copying to/from the USB memory:

```
"\hard disk\myFileName.txt"
```

Get

IND780 only

Reads a record from a random access file by record number into fields defined by the field statement.

Reads a record from the indexed sequential file into the fields defined by a FIELD statement. The program must first set a logical index into the key field of the FIELD

variables. The GET function executes a binary search of the file for a record containing the logical key. If it finds the record, the GET function returns the record in the FIELD variables. Otherwise, the GET function generates a "RECORD NOT FOUND" error. You must use an ON ERROR function to handle these errors.

Refer to the example provided in **IndexedFile.zip**.

Properties:

File/Serial IO Number	File number in "#<Filenum>" format.
Record Number	For random access files, the number of the record to be read. If a record number is not specified, the GET function returns the next sequential record.

For indexed-sequential files, the number is typically not specified. When it is not specified, the GET function returns the record specified in the keyword field of the FIELD statement. The INDEXED or SORTREC functions specify which field is the keyword field. When the record number is specified, the GET function returns the specified record number. The record number can be variable or a constant.

Indexed

IND780 only

Identifies which field in the record is the index key. The OPEN FILE function must first be used to open a file as a random access file and define the record format using the FIELD command. The INDEXED function identifies the file as an indexed-sequential file.

Refer to the example provided in **IndexedFile.zip**.

Properties:

File/Serial IO Number	File number in "#<Filenum>" format.
Variable Name	Name of the FIELD variable that is the index key.

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.

Refer to the example provided in **FileCreation.zip**.

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.

Kill

IND780 only

The TaskExpert KILL command supports variable names as well as fixed program names as command arguments.

Example 1:

```
10 KILL "\storage card\TaskExpert\data\test1.dat"
```

Example 2:

```
10 FilePath$ = "\storage card\TaskExpert\data\test1.dat"
20 KILL FilePath$
```

Also refer to the example provided in **Kill.zip**.

LSET

IND780 only

Moves the value of an expression or variable into a field in a random-access file buffer in preparation for a PUT statement. LSET left-justifies the value of a string variable in the field.

Refer to the example provided in **IndexedFile.zip**.

Properties:

String Variable [String variable, Output/Result variable] Any string variable or a random-access file field defined in a FIELD statement.

String Expression [String] The left-justified version of the string variable.

Open File

Accesses a file. Files can be sequential, random, or indexed-sequential files stored on the terminal.

Note: The IND560 only supports sequential files.

Refer to the example provided in **FileCreation.zip**.

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. Random access and indexed-sequential files must be opened as RANDOM.

Note:APPEND creates a file if it doesn't already exist.

File Number File number in "#<Filenum>" format. Valid File Numbers are #0 to #7.

Length [Integer value only, Optional] This is the record length for random access files and indexed-sequential files.

Example 1, accessing file on the Compact Flash card:

```
"\storage card\myFileName.txt"
```

Example 2, accessing file on USB memory: (USB option on IND780 only)

```
"\hard disk\myFileName.txt"
```

Print

Outputs data to the specified serial port or writes data to a sequential file. Refer to the example provided in **FileCreation.zip**.

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 semi-colons. 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.

Put

IND780 only.

Writes records to a random access file.

Writes a record to the indexed-sequential file. The program must first set values into the FIELD variables, including the logical key variable. The PUT statement searches the file for a record containing the logical key. If it finds the record, the PUT statement overwrites the existing record with the new data. If there is no existing record with the same key, the PUT statement inserts a new record into the file in its proper sequential position.

Field variables are cleared after the PUT statement is run.

Refer to the example provided in **IndexedFile.zip**.

Properties:

File/Serial IO Number	File number in "#<Filenum>" format.
Record Number	[Integer, Optional] Number of the record to write. When a record number is not specified for random access files, TaskExpert writes to the record specified by the indexed field of the field variables. **NOTE:** Record Number is not used for indexed-sequential files.

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.

Refer to the example provided in **DataRead.zip**.

Properties:

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

Restore

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

Refer to the example provided in **DataRead.zip**.

Properties:

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

RSET

Moves the value of an expression or variable into a specified field in a random-access file buffer in preparation for a PUT statement. RSET also right-justifies the value of a string variable in the field variable.

Refer to the example provided in **IndexedFile.zip**.

Properties:

String Variable [String variable, Output/Result variable] Any string variable or a random-access file field defined in a FIELD statement.

String Expression [String] The right-justified version of the string variable.

SortRec

IND780 only.

Identifies the file as an indexed-sequential file. Identifies which field is the index filed. Sorts the file records in sequential order by key if necessary.

Refer to the example provided in **SortRec.zip**.

Properties:

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

Variable Name The FIELD variable used as the index key.

Swap

Exchanges the values of two variables.

Refer to the example provided in **Swap.zip**.

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.

Refer to the example provided in **FileCreation.zip**.

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.

Chapter 8 Standard Database Table Commands, IND780

Command Reference

Command	Usage
Add ID	Adds the value to an item associated with selected ID(s) in a table of the Standard Database Tables
Add Item	Adds the value to an item in selected row/s in a table of the Standard Database Tables.
Close Standard Database	Terminates access to Standard Database Tables.
Create Standard Table	Creates a new Standard Database Table and opens it for access within TaskExpert.
Delete ID	Deletes specific rows from a table in the Standard Database Tables, by ID.
Delete Row	Deletes specific rows from a table in the Standard Database Tables, by description or by entry number.
Execute SQL	Submit an SQL command to the SQL CE server for the database instance.
Get Row	Enables the TaskExpert application to retrieve rows from a database rowset, one at a time.
Next Row	Retrieves the next row from a rowset from the Standard Database Tables.
Open Tables	Opens the currently existing Standard Database Tables for access within TaskExpert.
Select Row/Select ID	Selects all rows or specific rows from a table in the Standard Database Tables, by description or by entry number.
Set Item/Set Item ID	Sets the value of an item in selected row/s in a table of the Standard Database Tables.
Set Row	Inserts a new row entry into a specific table in the Standard Database Tables.

Add ID / Add Item

Add the value to an item in selected row/s in a table of the Standard Database Tables by Short ID. When there are multiple selected rows, ADDTOITEM adds the result to the first row-column item and writes the item result back to all selected rows.

Refer to the example provided in **Standard Table.zip**.

Properties:

StandardTable Tables A0 – A9.

Short ID (Add ID) [String] ShortID column for 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 SQL CE modifies the column value in all selected rows.

Record ID (Add Item) [String] Record ID for selected row. This number is unique for each row so this function will select only a single row.

Item [Integer] Data field 1 – 17 in selected row/s that you are modifying.

Data The data value you are adding to the selected row-column item. TaskExpert automatically converts both data values to numeric values before doing the addition and then converts it back to a string before writing it to the database table.

Status [Integer variable, output/result variable, Optional]
0 = Success, 1 = Failed

The ADDTOITEM function is a Table function operating only on the Standard Database Tables. It has a simpler syntax, but less flexibility, than the equivalent SQL commands. You cannot use this function with Custom SQL CE Databases.

Close Standard Database

Terminates access to Standard Database Tables.

Refer to the example provided in **Standard Table.zip**.

Create Standard Table

Create a new Standard Database Table and open it for access within TaskExpert.

This command creates the ten data tables A0 – A9 in their defined format and creates indexes for the entry # and description columns. The Standard Database file resides in the Compact Flash. It is always database instance #1.

Each table has 20 columns with the following column names and 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(16)
data10          NVARCHAR(16)
data11          NVARCHAR(16)
data12          NVARCHAR(16)
data13          NVARCHAR(40)
data14          NVARCHAR(40)
data15          NVARCHAR(40)
data16          NVARCHAR(40)
data17          NVARCHAR(40)
```

CREATE STANDARD TABLE generates an index on 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 Database Tables. These functions have a simpler syntax, but less flexibility, than the equivalent SQL commands. However, you can also use the SQL commands to manipulate the Standard Database Tables.

Refer to the example provided in **Create Standard Table.zip**.

Delete ID / Delete Row

Delete specific rows by description or by entry number from a table in the Standard Database Tables by Short ID (Delete ID) or by Record ID (Delete Row). When a row is deleted, SQL CE does not reuse the entry number associated with the deleted row.

Refer to the example provided in **StandardTable.zip**.

Properties:

StandardTable Tables A0 – A9.

Short ID (Delete ID) [String] Short ID column for the selected row/s. The ShortID column is not necessarily unique for each row so this function can delete multiple rows.

Record ID (Delete Row) [String] Record ID for the selected row. This number is unique for each row so this function can delete multiple rows. The Entry Number is unique for each row so this function will delete at most one row.

Status	[Integer variable, output/Result variable, Optional] 0 = Success, 1 = Failed

This is a Table function operating only on the Standard Database Tables. It has a simpler syntax, but less flexibility, than the equivalent SQL commands, and it cannot be used with Custom SQL CE Databases. SQL commands must be used for operations on these tables.

The DELETE ROW function is a Table function operating only on the Standard Database Tables. It has a simpler syntax, but less flexibility, than the equivalent SQL commands.

Execute SQL

Submit an SQL command to the SQL CE Server for the database instance.

TaskExpert supports the following SQL commands for both Custom Databases and Standard Database Tables:

Create Table	Create a table within the database and define the column names within the table.
Create Index	Create a key index for a particular column. An index enables the user application to quickly access a row entry from a table using a key value contained in the index column. You can define simultaneous indexes on different columns in the table.
Select	Retrieve one or more rows from a table.
Update	Update selected columns in existing rows with specific values for the columns.
Insert	Insert a new row into the table.
Delete	Delete a row from the table.
Drop Index	Delete an index for a table.
Drop Table	Delete a table from the database.

TaskExpert Supports the following SQL data types:

INT	Integer Number
NCHAR	Fixed-length String
NVARCHAR	Varying-length String
FLOAT	Double Floating Point Number

Please refer to a Windows CE SQL Server Command reference for a detailed description of the SQL commands. There is one later in this document.

The TaskExpert automatically applies the SQL CREATE TABLE and CREATE INDEX commands for the Standard Database Tables during execution of the CREATE STANDARD TABLE commands. You should not apply these commands to the Standard Database Tables.

Refer to the example provided in **ExecuteSQL.zip**.

Embedding Variables in SQL Commands

TaskGlobal variables and expressions can be embedded in an SQL command by enclosing the expression with curly braces. The TaskExpert interpreter resolves the variables and expressions within the SQL commands before submitting them to the SQL CE Server. Note in Example 2 that Shared Data Variables are passed to TaskGlobals before being written to the SQL statement. The EXECUTE SQL command does not accept passing Shared Data Variables or local variables into the statement. TaskGlobals *must be used* within the SQL statement.

Syntax:

```
SQL #n, SQL command {basic variable}, {basic expression}
```

Example 1 (query statement for a Standard Database Table):

```
state$ = ""
...
state$ = "Missouri"
SQL #1, SELECT * FROM A6 WHERE data2 = {state$}
```

Example 2 (inserting data into a Custom Database):

```
defshr SDweight$,wt0101
defshr SDrate#,wt0114
weight$ = ""
rate# = 0.0
...
for i% = 1 to 120
weight# = SDweight$
rate# = SDrate#
Sleep 100
SQL #4,INSERT WtTable VALUES( {i%}, {LTRIM$(weight$)}, {rate#} )
next i%
SQL #4,SELECT * FROM WtTable WHERE Rate > '100.0'
```

Get Row

After a SELECTROW command selects a "rowset from a Standard Database Table, or a SQL SELECT command selects a "rowset" from a Custom SQL CE Database or the Standard Database Tables, the GET ROW 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

Properties:

Instance Number "#n" is the instance number of the database. You may have up to four databases, including the Standard Database Tables, open concurrently. The legal values for #n are #1 through #4.

Value List of one or more variables to retrieve data from table. Items must be separated by commas. See syntax below.

Syntax:

numCol%, column1, column2,... columnN

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

The GET ROW command copies data from the retrieved row into variables column1 through columnN. The type of the GET ROW 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, the TaskExpert copies data up to the lesser of these two. It returns the number of columns copied in the numCoL% variable.

Refer to the example provided in **Select.zip**.

Next Row

Retrieve the next row from a rowset from the Standard Database Tables. The SELECT ROW function returns the first selected row from the table. Use the NEXT ROW table function to retrieve the subsequent rows.

Properties:

No 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.

The return values, variables, and Shared Data fields are the same as in the SELECT ROW function.

The NEXT ROW function is a Table function operating only on the Standard Database Tables. It has a simpler syntax, but less flexibility, than the equivalent SQL commands.

Refer to the example provided in **StandardTable.zip**.

Open Tables

Open the currently existing Standard Database Tables for access within TaskExpert.

The Standard Database Tables always resides in Compact Flash. It has ten tables A0 – A9. It is always instance #1. The CREATE STANDARD TABLE command establishes the format of the tables

Refer to the example provided in **StandardTable.zip**.

Select ID / Select Row

Select all rows or specific rows by description or by entry number from a table in the Standard Database Tables by Record ID. The SELECT ROW function also returns the first selected row from the table. Use the NEXT ROW Table function or the GET ROW command to retrieve the subsequent rows.

Refer to the example provided in **StandardTable.zip**.

Properties:

StandardTable	Tables A0 – A9.
Short ID (Select ID)	[String, Optional] Short ID column for the selected row/s. The ShortID column is not necessarily unique for each row so this function can select multiple rows.
Record ID (Select Row)	[String, Optional] Short ID column for the selected row. This number is unique for each row so this function will select at most one row.
Number of columns	This function returns the number of columns successfully retrieved in first selected row. If the return value is zero, there is no row selected.

TaskExpert variables:

The SELECT ROW 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 20 columns, the TaskExpert Interpreter sets the unused variables to a null value.

recID$	Column 1
shortID$	Column 2
description$	Column 3
data1$	Column 4
data2$	Column 5
data3$	Column 6
data4$	Column 7
data5$	Column 8
data6$	Column 9
data7$	Column 10
data8$	Column 11
data9$	Column 12
data10$	Column 13
data11$	Column 14
data12$	Column 15
data13$	Column 16
data14$	Column 17
data15$	Column 18
data16$	Column 19
data17$	Column 20

Shared Data fields:

The SELECT ROW 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 IND780 can automatically insert the database values into a print ticket or report.

dd--01	Column 1
dd--02	Column 2
dd--03	Column 3
dd--04	Column 4
dd--05	Column 5

dd--06	Column 6
dd--07	Column 7
dd--08	Column 8
dd--09	Column 9
dd--10	Column 10
dd--11	Column 11
dd--12	Column 12
dd--13	Column 13
dd--14	Column 14
dd--15	Column 15
dd--16	Column 16
dd--17	Column 17
dd--18	Column 18
dd--19	Column 19
dd--20	Column 20

The SELECT ROW function is a Table function operating only on the Standard Database Tables. It has a simpler syntax, but less flexibility, than the equivalent SQL commands. You cannot use this function with Custom SQL CE Databases.

Set Item / Set Item ID

Set the value of an item in selected row/s in a table of the Standard Database Tables by Record ID. When there are multiple selected rows, these commands write the item to all selected rows.

Refer to the example provided in **StandardTable.zip**.

Properties:

StandardTable	Tables A0 – A9.
Record ID (Set Item)	[String] Record ID for the selected row. This number is unique for each row so this function will select at most one row.
Short ID (Set Item ID)	[String] Short ID column for the selected row/s. The Short ID column is not necessarily unique for each row so this function can select multiple rows.
Item	[Integer] Data field 1 – 17 in selected row/s that you are modifying. 0 represents the description field.
Data	The data value you are inserting into the selected row-column item. TaskExpert automatically converts the data value to a string before inserting it into the database table.
Status	[Integer variable, Output/Result variable, Optional] 0 = Success, 1 = Failed

The SET ITEM function is a table function operating only on the Standard Database Tables. It has a simpler syntax, but less flexibility, than the equivalent SQL commands. You cannot use this function with Custom SQL CE Databases.

Set Row

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

Refer to the example provided in **StandardTable.zip**.

Properties:

table%	Tables A0 – A9, *required.*
shortID$	contents of short ID field, *required.*
description$	Contents of the description column in the row, *required.*
data 1	Contents of the 1st data column, *required.*
data 2	Contents of the 2nd data column, optional.
data 3	Contents of the 3rd data column, optional.
data 4	Contents of the 4th data column, optional.
data 5	Contents of the 5th data column, optional.
data 6	Contents of the 6th data column, optional.
data 7	Contents of the 7th data column, optional.
data 8	Contents of the 8th data column, optional.
data 9	Contents of the 9th data column, optional.
data 10	Contents of the 10th data column, optional.
data 11	Contents of the 11th data column, optional.
data 12	Contents of the 12th data column, optional.
data 13	Contents of the 13th data column, optional.
data 14	Contents of the 14th data column, optional.
data 15	Contents of the 15th data column, optional.
data 16	Contents of the 16th data column, optional.
data 17	Contents of the 17th data column, optional.
Status	[Integer variable, Output/Result variable, Optional] 0 = Success, 1 = Failed

The SET ROW command automatically converts the parameter data to string data, if necessary, before submitting it to the SQL CE Server to write to the database.

The SQL CE Server automatically generates an entry number for the new row. It uses the next successive number. Since the entry number is the Primary Key for the table, this guarantees that the entry number will be unique for each row. If you delete an existing entry, that entry number becomes unused and the SQL CE Server does not reuse it.

The SET ROW function is a Table function operating only on the Standard Database Tables. It has a simpler syntax, but less flexibility, than the equivalent SQL commands. You cannot use this function with Custom SQL CE Databases. You must use the SQL commands for operations on these databases.

Chapter 9 Standard Table Commands, IND560

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.

Create Standard Tables

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(16)
data10       NVARCHAR(16)
data11       NVARCHAR(16)
data12       NVARCHAR(16)
```

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.

Refer to the example provided in **StandardTables.zip**.

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.
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 $.

Open Tables

Open the currently existing Standard Tables for access within TaskExpert.

Refer to the example provided in **StandardTables.zip**.

Properties:

Standard Table	Tables A0 – A9.

Close Standard Tables

Terminate access to Standard Tables.

Refer to the example provided in **StandardTables.zip**.

Clear or Delete Standard Tables

Clear or Delete standard tables.

Refer to the example provided in **StandardTables.zip**.

Properties:

Standard Table	Tables A0 – A9.
Action	Clear or Delete

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.

Refer to the example provided in **Select.zip**.

Set Row

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

Refer to the example provided in **StandardTables.zip**.

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.

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.

Refer to the example provided in **StandardTables.zip**.

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.

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.

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

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 IND560 can automatically insert the table values into a print ticket or report.

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

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.

Refer to the example provided in **StandardTables.zip**.

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.

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.

Refer to the example provided in **StandardTables.zip**.

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`

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.

Refer to the example provided in **StandardTables.zip**.

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`

Delete Row

Delete specific rows by description 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.

Refer to the example provided in **StandardTables.zip**.

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`

Select

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

Refer to the example provided in **Select.zip**.

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.

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
```

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
```

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
```

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
```

Chapter 10 Custom Database Commands, IND780

Command Reference

Custom Database Commands

Command	Usage
Close Database	Terminates access to a Custom Database.
Create Database	Creates a new Custom Database and opens it for access within TaskExpert.
Get Row	Selects a "rowset" from a Custom SQL CE Database or the Standard Database Tables
Execute SQL	Submits an SQL command to the SQL CE Server.
Open Database	Opens an existing Custom Database for access within TaskExpert.

Close Database

Terminates access to a Custom Database.

Refer to the example provided in **Custom Database.zip**.

Create Database

Creates a new custom database and open it for access within TaskExpert.

Refer to the example provided in **Custom Database.zip**.

Properties:

Instance Number "#n" is the instance number of the database. You may have up to four databases, including the Standard Database Tables, open concurrently. The legal values for #n are #1 through #4.

File Name The "filename" is the file in the Windows CE file system where the database resides. The file may reside in Compact Flash or in Dynamic RAM. Enclose the file directory in quotes as seen below.

Example:

"\storage card\taskexpert\data\customDBase.sdf"

Execute SQL

Submit an SQL command to the SQL CE Server for the database instance.

TaskExpert supports the following SQL commands for both Custom Databases and Standard Database Tables:

CREATE TABLE	Create a table within the database and define the column names within the table.
CREATE INDEX	Create a key index for a particular column. An index enables the user application access quickly a row entry from a table using a key value contained in the index column. You can define simultaneous indexes on different columns in the table.
SELECT	Retrieve one or more rows from a table.
UPDATE	Update selected columns in existing rows with specific values for the columns.
INSERT	Insert a new row into the table.
DELETE	Delete a row from the table.
DROP INDEX	Delete an index for a table.
DROP TABLE	Delete a table from the database.

TaskExpert Supports the following SQL data types:

INT	Integer Number
NCHAR	Fixed-length String
NVARCHAR	Varying-length String
FLOAT	Double Floating Point Number

Please refer to a Windows CE SQL Server Command reference for a detailed description of the SQL commands. There is one later in this document.

The TaskExpert automatically applies the SQL CREATE TABLE and CREATE INDEX commands for the Standard Database Tables during execution of the CREATE STANDARD TABLE and CREATE JOINED TABLES commands. You should not apply these commands to the Standard Database Tables.

Get Row

After a SELECTROW command selects a "rowset from a Standard Database Table, or a SQL SELECT command selects a "rowset" from a Custom SQL CE Database or the Standard Database Tables, the GET ROW 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

Properties:

Instance Number	"#n" is the instance number of the database. You may have up to four databases, including the Standard Database Tables, open concurrently. The legal values for #n are #1 through #4.

Value List of one or more variables to retrieve data from table. Items must be separated by commas. See syntax below.

Syntax:

numCol%, column1, column2,... columnN

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

The GET ROW command copies data from the retrieved row into variables column1 through columnN. The type of the GET ROW 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, the TaskExpert copies data up to the lesser of these two. It returns the number of columns copied in the numCoL% variable.

Refer to the example provided in **Select.zip**.

Open Database

Open an existing custom database for access within TaskExpert.

Refer to the example provided in **Select.zip**.

Properties:

Instance Number "#n" is the instance number of the database. You may have up to four databases, including the Standard Database Tables, open concurrently. The legal values for #n are #1 through #4.

File Name The "filename" is the file in the Windows CE file system where the database resides. The file may reside in Compact Flash or in Dynamic RAM. Enclose the file directory in quotes as seen below.

Example:

"\storage card\taskexpert\data\customDBase.sdf"

Chapter 11 Remote Database Commands, IND780

Command Reference

Remote Terminal Database Commands

These commands are used for communication within a cluster of IND780s.

Command	Usage
Add Remote Item	Add the value to an item in a selected row(s) in a table of the Remote Standard Database Tables.
Add to Remote ID	
Close Cluster Remote Standard Database	Terminate access to Remote Standard Database Tables.
Create Cluster Remote Standard Database	Create new Remote Standard Database Tables and open the database for access within TaskExpert from a remote cluster terminal.
Delete Remote ID	Delete specific rows by Short ID or Record ID from a table in the Cluster Remote Standard Database Tables.
Delete Remote Row	
Next Remote Row	Retrieve the next row from a rowset from the Cluster Remote Standard Database Tables.
Open Cluster Remote Standard Database	Open the currently existing Remote Standard Database Tables for access within TaskExpert.
Select Remote Row	Select all rows or specific rows by Short ID or Record ID from a table in the Cluster Remote Standard Database Tables.
SelRemID	
Set Remote Item	Set the value of an item in a selected row(s) in a table of the Remote Standard Database Tables.
SetRemID	
Set Remote Row	Insert a new row entry into a specific table in the Cluster Remote Standard Database Tables.

Remote PC Database Commands

These commands are used for communication between an IND780 and SQL database located on a remote PC.

Command	Usage
GetSQLErr	Retrieves an error string from the SQL Client engine.
GetSQLRes	Retrieves the system HRESULT error code of the last RDA command executed.
RDAPull	Extracts data from a SQL Server database table and stores it in a SQL Server CE database table.
RDAPush	Transmits data to a SQL Server database table from a SQL Server CE database table.
RDAServer	Sets up the data connection to the PC SQL Server and Server Agent.
RDASQLExe	Submits SQL statements to execute on the SQL Server.
Replication Add	Prepares a local database for use in replication.

Command	Usage
Replication Drop	Removes replication information from a local database
Replication Sync	Merges the changed local records with SQL Server and read the SQL Server records into the local tables as defined by the publication.
Replint	Initializes the replication fields before a synchronization.

Remote Terminal (Cluster) Database Commands

Add Remote Item / Add to Remote ID

Adds the value to an item in selected row(s) in one of the Remote Standard Database Tables. When multiple rows are selected, `ADDTORITM` adds the result to the first row-column item and writes the item result back to all selected rows.

Note: The clustered database is always Instance #2.

Properties:

Table — Tables A0 – A9

Record ID (Add to Remote Item) — [String] Record ID for the selected row. This value is unique for each row so this function will select at most one row.

ShortID (Add to Remote 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. If the shortID selects multiple rows, the SQL CE modifies the column value in all selected rows.

Item — [Integer] Data field 1 – 17 in selected row(s) to be modified.

Data — The data value to be added to the selected row-column item. TaskExpert automatically converts both data values to numeric values before doing the addition and then converts them back to a string, which it then writes to the database table.

Status — 0 = SUCCESS, 1 = FAILED

Delete Remote Row / Delete Remote ID

Delete specific rows by description or by entry number from a table in the Cluster Remote Standard Database Tables. When a row is deleted, SQL CE does not reuse the entry number associated with the deleted row.

Note: The clustered database is always Instance #2.

Properties:

Table	Tables A0 – A9
Record ID (Delete Remote Row)	[String] Record ID for the selected row. This value is unique for each row so this function will select at most one row.
ShortID (Delete Remote 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. If the shortID selects multiple rows, the SQL CE modifies the column value in all selected rows.
Return Value	0 = SUCCESS, 1 = FAILED

Close Cluster Remote Standard Database

Terminates access to Remote Standard Database Tables.

Note: The clustered database is always Instance #2.

Create Cluster Remote Standard Database

Creates new Remote Standard Database Tables and opens the database for access to it, within TaskExpert, from a remote cluster terminal. The command creates the ten data tables, A0 to A9, at the node addressed, each in its defined format.

The Remote Standard Database file resides in the Compact Flash at the designated cluster node. It is always database instance #2.

The only SQL command that will work on remote standard database tables is SELECT.

Note: The clustered database is always Instance #2.

Properties:

Node	[Integer] 1 – 20 indicating node in the cluster; required.

Next Remote Row

Retrieves the next row from a rowset from the Cluster Remote Standard Database Tables. The SELREMROW function returns the first selected row from the table. Use the NXTREMROW table function to retrieve the subsequent rows or rows created by SQL SELECT command.

This command operates only on Cluster Remote Standard Database tables.

The return values, TaskExpert variables, and Shared Data fields are the same as in the SELREMROW function – refer to **Select Remote Row / SelRemID**, below.

Note: The clustered database is always Instance #2.

Open Cluster Remote Standard Database

Opens the currently existing Remote Standard Database Tables for access within TaskExpert.

Note: The clustered database is always Instance #2.

Properties:

Node [Integer] 1 – 20 indicating node in the cluster; required.

Select Remote Row / SelRemID

Retrieves the next row from a rowset from the Cluster Remote Standard Database Tables. The SELREMROW function returns the first selected row from the table. Use the NXTREMROW table function to retrieve the subsequent rows or rows created by SQL SELECT command.

These commands operate only on Cluster Remote Standard Database tables.

Note: The clustered database is always Instance #2.

Properties:

Table Tables A0 – A9

Record ID (Select Remote Row) [String] Record ID for the selected row. This value is unique for each row so this function will select at most one row.

ShortID (SelRemID) [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 SQL CE modifies the column value in all selected rows.

Return Value Number of columnsThis function returns the number of columns successfully retrieved in first selected row. If the return value is zero, there is no row selected.

TaskExpert variables

The `SELREMID / SELREMROW` 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 20 columns, the TaskExpert Interpreter sets the unused variables to a null value.

recID	Column 1
shortID	Column 2
description	Column 3
data 1 to 17	Columns 4 to 20

Shared Data Fields The SELREMROW 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 IND780 can automatically insert the database values into a print ticket or report.

dd--01	Column 1
to	
dd--20	Column 20

Set Remote Item / SetRemID

Sets the value of an item in a selected row(s) in a table of the Remote Standard Database Tables. When multiple rows are selected, `SETREMITM` writes the item to all selected rows.

These commands operate only on Cluster Remote Standard Database tables.

Note: The clustered database is always Instance #2.

Properties:

Table Tables A0 – A9

Record ID (Set Remote Item) [String] Record ID for the selected row. This value is unique for each row so this function will select at most one row.

ShortID (SetRemID)	[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 SQL CE modifies the column value in all selected rows.
Item	[Integer] Data field 1 – 17 in selected row(s) to be modified. 0 represents the description field.
Data	The data value to be inserted into the selected row-column item. TaskExpert automatically converts the data value to a string before inserting it into the database table.
Return Value	0 = SUCCESS, 1 = FAILED

Set Remote Row

Inserts a new row entry into a specific table in the Cluster Remote Standard Database Tables.

The SETREMROW command automatically converts the parameter data to string data, if necessary, before submitting it to the SQL CE Server to be written to the database.

The SQL CE Server automatically generates a record ID (unique identifier or GUID) for the new row. Since the record ID is the Primary Key for the table, this will be unique for each row.

This command operates only on Cluster Remote Standard Database tables.

Note: The clustered database is always Instance #2.

Properties:

Table	Tables A0 – A9; required
ShortID	[String] Contents of short ID field; required
Description	[String] Contents of the description column in the row; required
Data1	Contents of the 1st data column; required
Data2 to Data 17	Contents of the 2nd to 17th data columns; optional
Return value	0 = SUCCESS, 1 = FAILED

Remote PC Database Commands

Remote Data Access (RDA)

Using the Remote Data Access (RDA) feature in the SQL Server CE, a TaskExpert application can access (pull) data from a remote SQL Server database table and

store that data in a local SQL Server CE database table. The application can then read and update the local SQL Server CE database table. SQL Server CE can optionally track all changes that the application makes to the local table. The application can later update (push) the changed records from the local table back to the SQL Server table.

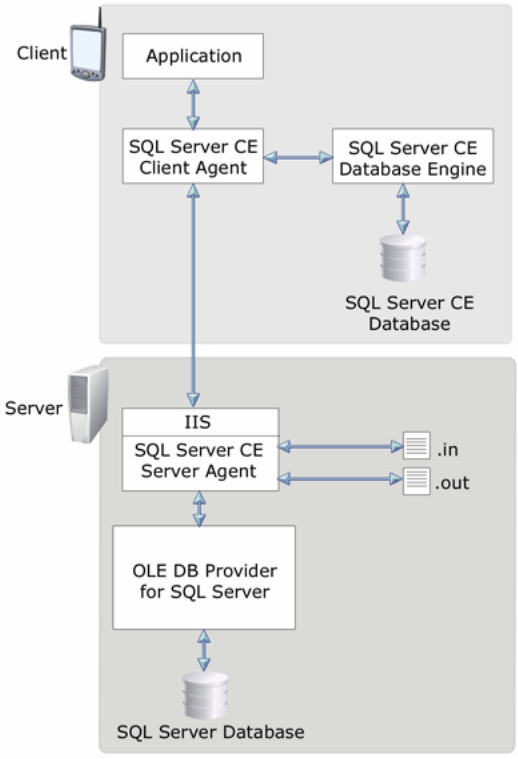
TaskExpert applications can also use RDA to submit SQL statements for execution on a remote SQL Server database. For example, an application can submit SQL statements to a remote SQL Server table, that insert, update or delete records. Applications can invoke any SQL statement that does not return a record set.

SQL Server CE communicates with SQL Server through Microsoft Internet Information Services (IIS). By connecting through IIS, RDA takes advantage of IIS authentication and authorization services. In the IND780, RDA operates over the TCP/IP local area network (LANs).

RDA Architecture

Remote Data Access (RDA) uses the SQL Server CE Database Engine, SQL Server CE Client Agent, and SQL Server CE Server Agent.

The following illustration shows how these components work together.



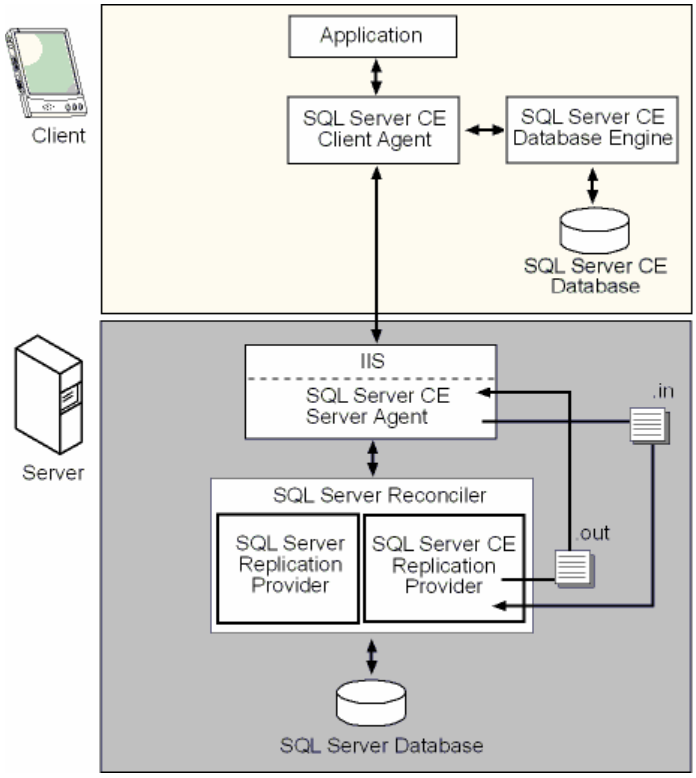
The SQL Server CE Database Engine manages the SQL Server data store on the IND780. For pull operations that are tracked, the Database Engine tracks all database records that are inserted, updated, or deleted by maintaining a small amount of change tracking information with each record. If indexes exist on the SQL Server data, RDA also supports creating indexes on the local data.

SQL Server CE Client Agent is the primary RDA component on the IND780. It implements the RDA object interface. The TaskExpert Interpreter calls this interface so that the TaskExpert applications can programmatically control the RDA Pull, Push, and SubmitSQL operations.

SQL Server CE Server Agent runs on the Database Server PC; it is responsible for handling the HTTP requests made by SQL Server CE Client Agent.

Replication Architecture

Replication makes use of the SQL Server CE Database Engine and the SQL Server CE Client Agent on the IND780; and the SQL Server CE Server Agent and SQL Server CE Replication Provider on the Windows PC SQL Server.



SQL Server CE Database Engine

The SQL Server CE Database Engine manages the SQL Server data store on the IND780 Windows CE device. For subscription tables, the Database Engine tracks all records that the client inserts, updates, or deletes by maintaining a small amount of change tracking information with each record.

SQL Server CE Client Agent

The SQL Server CE Client Agent is the primary SQL Server CE replication component on the IND780 Windows CE device. It implements the SQL Server CE Replication Object interface. Applications must call this interface to control programmatically replication.

When the IND780 Task Expert application initiates synchronization using the ReplSync function, the SQL Server CE Client Agent extracts all inserted, updated, and deleted records from the SQL Server CE Subscriber and propagates them to the SQL Server CE Server Agent through HTTP.

Conversely, when the SQL Server CE Server Agent propagates data changes at the Publisher back to the IND780 Windows CE device, the SQL Server CE Client Agent applies these changes to the SQL Server CE subscription database on the IND780.

SQL Server CE Server Agent

The SQL Server CE Server Agent is an ISAPI DLL. It is responsible for handling the HTTP requests made by the SQL Server CE Client Agent. When synchronization occurs, the SQL Server CE Server Agent creates a new input message file on the IIS server and writes the data insert, update, and delete requests that the IND780 SQL Server CE Client Agent sends into that file. After writing all data insert, update, and delete requests to the input message file, the IND780 SQL Server CE Server Agent initiates the SQL Server Reconciler process and waits for it to complete.

When the SQL Server Reconciler process is completed, the SQL Server CE Server Agent locates the output message file created by the SQL Server CE Replication Provider. This file contains the changes that have occurred at the publisher, and Replication must apply this file to the subscription database on the IND780 Windows CE device. The SQL Server CE Server Agent reads the output message file and transmits it to the SQL Server CE Client Agent on the IND780 Windows CE device.

SQL Server CE Replication Provider

The SQL Server Reconciler invokes the SQL Server CE Replication Provider during database synchronization. The SQL Server CE Replication Provider reads the input message file to inform the SQL Server Reconciler about changes made to the SQL Server CE subscription database that it must apply to the publication. The SQL Server Reconciler, in turn, informs the SQL Server CE Replication Provider about changes made at the Publisher that the Windows CE device must apply to the subscription database. The SQL Server CE Replication Provider writes these changes to an output message file it creates on the IIS system.

GetSQLErr

Retrieves an error string from the SQL Client engine. This command should be called if an RDA function returns with an error code of -1 through -3.

Properties:

Idx	[Integer] Index value or 1 to 3 of the error string requested.
Return Values	Error number and Error string, in the format ErrNum: nnnnn, ErrStr: aaaaa . . . . . aaaaa

Example return errors:

```
ErrNum: 28574, ErrStr: An internal error occurred. [ID,,,,,]
ErrNum: 00017, ErrStr: [DBNETLIB][ConnectionOpen
(Connect()).]SQL Server does not exist or access denied.
```

GetSQLRes

Remote PC DB GetSQLRes

Retrieves the system HRESULT error code of the last RDA command executed.

Properties:

Res [Integer] Error code.

Return Values

`0`	`No error`

Any non-zero value indicates the HRESULT error code from the system operation.

RDAPull

Remote PC DB RDAPull

Note: The SQL "DROP TABLE" command must be executed in TaskExpert before calling this function.

Extracts data from a SQL Server database table and stores it in a SQL Server CE database table. The RDAPull function essentially creates an image of the SQL Server table, or selected records table, in the local database.

Properties:

Local Table [String] The name of the SQL Server table to pull into the local database.

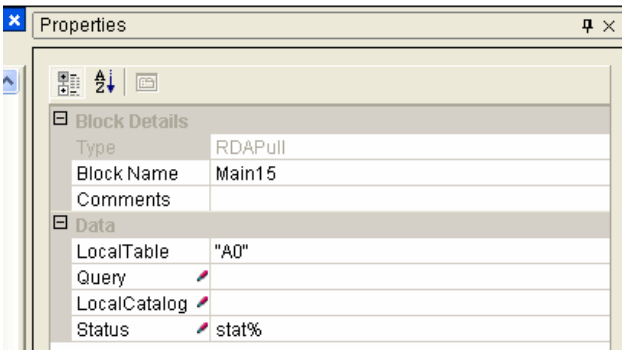
Query [String] Optional parameter to select specific records to be pulled to the local table (eg. "SELECT * FROM A8 WHERE data5 = 'SCL2'").

LocalCatalog [String] Name of the local database into which the table should be pulled. This parameter is optional; if it is absent, the standard database is the default.

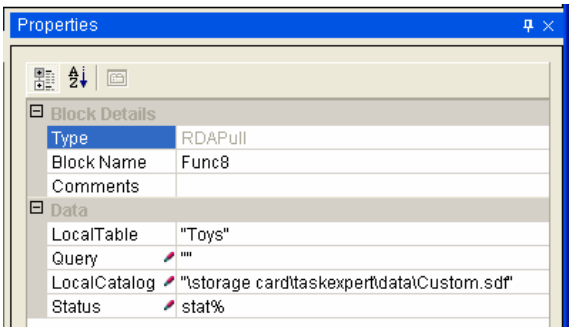
Return Values

`0`	`Successfully executed`
`-1 to -3`	`The number of errors available to read using GetSQLErr().`
`-99`	`Insufficient information to execute the function`

The following images show examples of properties for a Standard Database Table Pull and Custom Database Table Pull, respectively.



Standard Database Table Pull Properties



Custom Database Table Pull Properties

RDAPush

Note: A table can only be "pushed" if an RDAPull function has created it.

Transmits data to a SQL Server database table from a SQL Server CE database table. The RDAPush function will push the records to the SQL Server database table that the TaskExpert application has added, modified or deleted.

Properties:

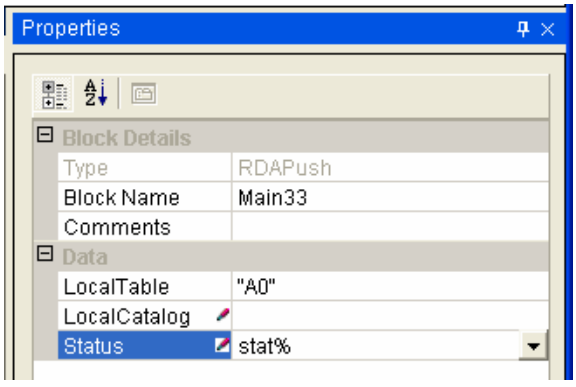
Local Table [String] The name of the SQL Server table to pull into the local database.

LocalCatalog [String] Name of the local database into which the table should be pulled. This parameter is optional; if it is absent, the standard database is the default.

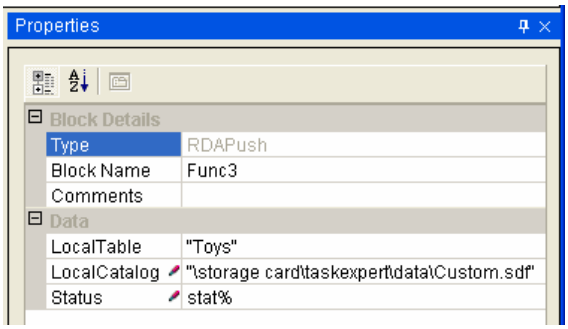
Return Values

`0`	`Successfully executed`
`-1 to -3`	`The number of errors negated available to read using GetSQLErr().`
`-99`	`Insufficient information to execute the function`

The following images show examples of properties for a Standard Database Table Push and Custom Database Table Push, respectively.



Standard Database Table Push Properties



Custom Database Table Push Properties

RDAServer

Note: This command must be executed before an RDAPull or RDApush is issued.

Sets up the data connection to the PC SQL Server and Server Agent.

Properties:

ServerAgent [String] The URL of the SQL Server CE Agent on the server computer.

Note: The URL can contain either the machine name or a direct IP address for the server address.

ServerLogin [String] The Network login name.

ServerPwd [String] The Network login password.

SqlServer [String] The SQL Server name.

SqlCatalog [String] The SQL Server database containing the table/s.

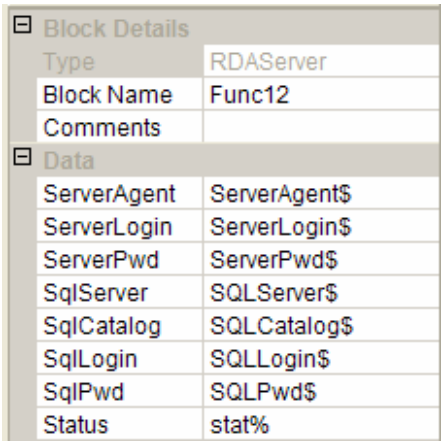
SqlLogin [String] The SQL Server database login name.

SqlPwd [String] The SQL Server database login password.

Return Values

```
0              Successfully executed
```

The following images show an example of the RDA Server block details and data, and an example of values used for the command, respectively.



```
ServerAgent$ = "http://172.18.54.100/CEagentRDA1/sscesa20.dll"
ServerLogin$ = "MTNA"
ServerPwd$ = "admin"
SQLServer$ = "MTNATESTPC\RDATEST"
SQLCatalog$ = "780RDA"
SQLLogin$ = "MTNA"
SQLPwd$ = "admin"
```

RDASQLExe

Note: RDA only supports SQL statements that do not return a record set.

Submits SQL statements to execute on the SQL Server.

Properties:

Query [String] A valid SQL statement to be executed.

Return Values

`0`	`Successfully executed`
`-1 to -3`	`The number of errors negated that are available to be read using GetSQLErr`
`-99`	`Insufficient information to execute the function`

Example syntax:

```
DELETE A2 WHERE shortID = '111'
```

Replication Add

This is the Replication function used to prepare to create a custom local database from a publication on the SQL Server. This must be called before the first ReplSync is called. This is not needed for existing standard or custom databases, since ReplSync will add the replication fields to the table(s) automatically the first time a sync is performed.

Properties:

LocalCatalog [String] Name of the local database. This parameter is optional; if it is absent, the standard database is default.

CreateDB [Integer] Creates the database:

0 = Use existing database

1 = Create a new empty database

Return Values

`0`	`Successfully executed`
`-1 to -3`	`The number of errors negated that are available to be read using GetSQLErr`
`-99`	`Insufficient information to execute the function`

Replication Drop

Removes replication information from a local database. This is only required if replication will no longer be used with the database.

Properties:

LocalCatalog [String] Name of the local database. This parameter is optional; if it is absent, the standard database is default.

DropDB [Integer] Remove/delete the local database:

0 = Leave database intact

1 = Delete database

Return Values

`0`	`Successfully executed`
`-1 to -3`	`The number of errors that are available to be read using GetSQLErr`
`-99`	`Insufficient information to execute the function`

Replication Sync

Note: The Replinit function must be called before each Replication Sync function.

Merges changed local records with SQL Server and reads the SQL Server records into the local tables, as defined by the publication.

Properties:

LocalCatalog [String] Name of the local database containing the table to be synchronized. This parameter is optional; if it is absent, the standard database is default.

Return Values

`0`	`Successfully executed`
`1`	`A positive value indicates success with some conflicts. The return value is the number of conflicts`
`-1 thru -3`	`The number of errors that are available to read using GetSQLErr().`
`-99`	`Insufficient information to execute the function`

Replinit

Initializes the replication fields before a synchronization is performed.

Properties:

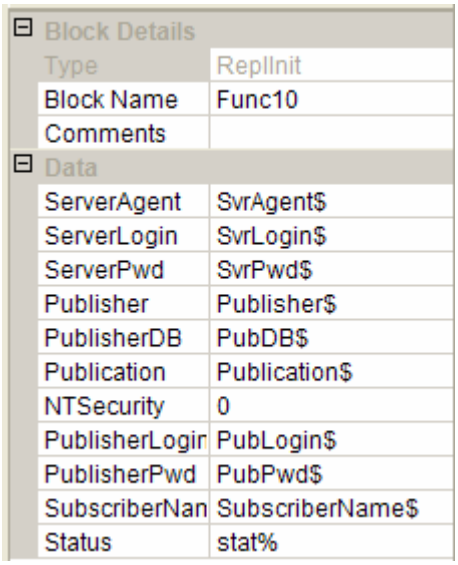
ServerAgent [String] The URL of the SQL Server CE Agent on the server computer.

ServerLogin [String] Network login name.

ServerPwd [String] Network login password.

Publisher	[String] SQL Server name.
PublisherDB	[String] SQL Server database providing the publication.
Publication	[String] Name of the publication.
NTSecurity	[String] 0 = Use DB Authentication 1 = Use NT Authentication
PublisherLogin	[String] SQL Server database login name.
PublisherPwd	[String] SQL Server database login password.
SubscriberName	[String]
Return Values	`0 Successfully executed`

The images below show an example of Block Details and Data for the Replinit function, and example values, respectively.



Block Details	
Type	ReplInit
Block Name	Func10
Comments	
Data	
ServerAgent	SvrAgent$
ServerLogin	SvrLogin$
ServerPwd	SvrPwd$
Publisher	Publisher$
PublisherDB	PubDB$
Publication	Publication$
NTSecurity	0
PublisherLogir	PubLogin$
PublisherPwd	PubPwd$
SubscriberNan	SubscriberName$
Status	stat%

```
SvrAgent$ = "http://172.18.54.150/CEagentRDA/sscesa20.dll"
SvrLogin$ = "US03W-SIMMONS-L\780test"
SvrPwd$ = "780test0001"
Publisher$ = "US03W-SIMMONS-L\IND780GROUP"
PubDB$ = "IND780"
Publication$ = "IND780Publication"
PubLogin$ = "earnest"
PubPwd$ = "earnest"
SubscriberName$ = "IND780"
```

Chapter 12 Miscellaneous Commands

Command Reference

Note: Except as indicated with an asterisk (*), commands are common to IND560 and IND780.

Note: For the **Node** property in common commands, the only legal value for the IND560 is 0.

Command	Usage
ADDTIME	Adds days, hours, minutes and years to a double floating point representation of date and time.
ArrayFind*	Search an array for a specific value.
ArrayCopy*	Copy the contents of one numeric array to another.
ClkTick	Returns the number of milliseconds since the last power up of the terminal.
Decrypt*	Decrypts cipher text and returns the plain text Unicode string.
Encrypt*	Encrypts the Unicode string "plaintext$" and stores the ciphered text data in the user-defined integer array.
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.
Language ID List*	Accesses the custom message database (IND780 only).
Option Base	Declares a minimum value for the numbers which can be used to access the elements of an array.
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.
Start Time	Starts the internal timer.
Stop Time	Stops the internal timer.
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.

ADDTIME

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

Refer to the example provided in **AddTime.zip**.

Properties:

OldTime [Double] Double floating-point representation of the date and time.

Days [Double] 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.

ArrayFind

Search an array for a specific value.

Refer to the example provided in **RecvArray.zip**.

Properties:

Array Name [String] Name of the TaskGlobal array.

Value [Integer] Value of the data to find.

Offset [Integer] Optional index location into the array to start the search.

Direction [Integer] Direction of the find where 0=forward and 1=reverse.

Forward increments through the array. Reverse decrements through the array. Forward is the default.

Type [Integer] Comparison to be performed where 0=equal and 1=not equal.

Equal means the values must be the same. Not equal means the values must not be the same. Not equal is useful to find the end of valid data in a reverse find.

Status `[Integer] Index to found value. -1 is returned if the comparison is not successful.`

ArrayCopy

Copy the contents of one numeric array to another.

Refer to the example provided in **RecvArray.zip**.

Properties:

Source Name	[String] Name of the TaskGlobal source array to copy data from.
Destination Name	[String] Name of the TaskGlobal 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.`

Clktick

This function allows precise timing loops and event timing.

Properties:

Clktick	[Double] Double floating-point representation of the time, in milliseconds, since the last power up of the terminal.

Decrypt

IND780 only.

Decrypts cipher text data and returns the plain text Unicode string.

Properties:

plaintext$ = Decrypt$("integer array name")

Return value:

Plain text Unicode string.

Note: Encrypt reads shared data ky0106 to determine which of the five cipher keys (ky0101 to ky0106) to use. The plain text string is encrypted using the OFB mode of AES (128 bit).

Encrypt

IND780 only.

Encrypts the Unicode string "plaintext$" and stores the ciphered text data in the user defined integer array. The first four bytes of the cipher text data are not encrypted and contain the length of the plaintext$ string in bytes (not characters).

Properties:

stat% = Encrypt(plaintext$, "integer array name")

Return values:

stat% returns the length of plaintext$ in bytes (not characters) if Encrypt is successful, else the following error codes are returned:

0 = Accompanies a syntax error.

-1 = Could not find or create the cipher text array.

-2 = Name of cipher text array exceeds 30 characters.

Note: Encrypt reads shared data ky0106 to determine which of the five cipher keys (ky0101 to ky0106) to use. The plain text string is encrypted using the OFB mode of AES (128 bit).

Get Task ID

Get Task ID of this running task.

Refer to the example provided in **MultiTasking.zip**.

Properties:

Task ID [Integer variable, Output/Result variable] Returns the task slot. Possible return values are 1 to 12.

JulDate

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

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.

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.

Example:

```
Datetime$ = "2009-01-29 09:55:23"
MyDatetime# = JulDate(Datetime$)
```

Output:

```
2009029.413466435
```

Language

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

Refer to the **IND780 Language Table Reference** for further details on the custom message database, and to the example provided in **Language.zip**.

Properties:

Index [Integer] Message index.

Text [String variable, Output/Result variable]
Returned message text.

Language ID List

IND780 only.

Refer to the **IND780 Language Table Reference** for further details on the custom message database, and to the example provided in **Language.zip**.

Properties:

ID List List of comma separated message IDs.

Output String [String variable, Output/Result variable]
Language list ID string.

Option Base

Declares a minimum value for array subscripts. Subscripts are the numbers which can be used to access the elements of an array. OPTION BASE gives an error if the base value is changed.

Refer to the example provided in **Option Base.zip**.

Properties:

Base Value Sets the lowest value any array subscript can have to 0 or 1. 1 is the default setting.

Resume Task

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

Refer to the example provided in **MultiTasking.zip**.

Properties:

Task ID [Integer] Selects the task slot. Legal values are 1 to 12.

Node [Integer, Optional] Selects the node in the cluster. Legal values are 0 & 1 to 20. 0 refers to the local node.

Status [Integer variable, Output/Result variable, Optional]

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.

Refer to the example provided in **Security.zip**.

Properties:

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

Start Task

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

Refer to the example provided in **MultiTasking.zip**.

Properties:

TaskID [Integer] Selects the task slot. Legal values are 1 to 12.

Node [Integer] Selects the node in the cluster. Legal values are 0 & 1 to 20. 0 refers to the local node.

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

Stop Task

Stop a selected TaskExpert application. Stop terminates the application.

Refer to the example provided in **MultiTasking.zip**.

Properties:

Task ID [Integer] Selects the task slot. Legal values are 1 to 12.

Node [Integer] Selects the node in the cluster. Legal values are 0 & 1 to 20. 0 refers to the local node.

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

Start Timer

Starts the internal timer. The maximum timer value is a long integer (2,147,483,647), representing approximately 2.1 million seconds.

Properties:

Milliseconds [Integer] The time, in milliseconds, to start the internal timer.

Stop Timer

Stops a running timer.

Suspend Task

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

Refer to the example provided in **MultiTasking.zip**.

Properties:

Task ID [Integer] Selects the task slot. Legal values are 1 to 12.

Node [Integer, Optional] Selects the node in the cluster. Legal values are 0 & 1 to 20. 0 refers to the local node.

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

TimDat

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

Refer to the example provided in **TimeDate.zip**.

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

Chapter 13 Display Objects 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.

Examples of each of these commands in use may be found in **Runtime.zip**.

Command Reference

Display Commands

Command		Usage
IND560	IND780	
Combobox		Displays Combobox on the screen in the application window.
Combobox and Label		Displays Combobox with a label on the screen in the application window.
Draw Image		Draws a graphic image in the application window.
Draw Shared Variable		Displays contents of Shared Data in the application window.
ImageSD		Draws a variable graphic image in the application window
Label		Displays a label on the screen in the application window.
Popup Display		Draws a POPUP Box in the application window.
Runtime Display		Creates unique application displays, including ability to set the size and number of weight displays, the size and type of SmartTrac display, the contents of the softkeys, and the contents of the entire display area.
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.

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.

Properties

Index
[Integer] The index of the display object in the application display. Legal values are 1 to 49. Object 50 is reserved for Single-Line Data Entry Line Prompt.

Note: The IND560 supports only 20 display objects. Indexes 1 to 19 are valid. Index 20 is reserved for the Data Entry Line.

Index Variable Instance
[Integer constant only, Optional] Index of the shared variables to be associated. Legal values are 1-49.

X-Coordinate
[Integer] Position of the text display relative to the top, left-hand corner of the application display window. Legal values = 1 to 320.

Y-Coordinate
[Integer] Position of the text display relative to the top, left-hand corner of the application display window. Legal values = 1 to 240.

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.

Select font from list: Arial (default) and Courier New in 10, 12, 14 and 16 point sizes.

The application can change the size of individual font using the SetFont function.

IND560 font options are Alpha, Chinese, Num1 and Num5. Please refer to **Appendix E**.

Color
[Optional] Select text color.

Options: Black (default), Red, Blue, Green, White, Orange, Yellow, Purple.

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 tx0150. 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.

Combobox and Label

Displays the Combo 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 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.

Properties

Index
[Integer] The index of the display object in the application display. Legal values are 1 to 49. Object 50 is reserved for Single-Line Data Entry Line Prompt.

Note: The IND560 supports only 20 display objects. Indexes 1 to 19 are valid. Index 20 is reserved for the Data Entry Line.

Index Variable Instance
[Integer constant only, Optional] Index of the shared variables to be associated. Legal values are 1-49.

X-Coordinate
[Integer] Position of the text display relative to the top, left-hand corner of the application display window. Legal values = 1 to 320.

Y-Coordinate
[Integer] Position of the text display relative to the top, left-hand corner of the application display window. Legal values = 1 to 240.

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.

Select font from list: Arial (default) and Courier New in 10, 12, 14 and 16 point sizes.

The application can change the size of individual font using the SetFont function.

IND560 font options are Alpha, Chinese, Num1 and Num5. Please refer to **Appendix E**.

Color
[Optional] Select text color.

Options: Black (default), Red, Blue, Green, White, Orange, Yellow, Purple.

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 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.

Label Index	[Integer] Index of the display object in the application display. Legal values are 1 to 49.
Label X-Coordinate	[Integer] X-Coordinate is position of the text display relative to the top, left-hand corner of the application display window. Legal values are 1 to 320.
Label Y-Coordinate	[Integer] Y-Coordinate is position of the text display relative to the top, left-hand corner of the application display window. Legal values are 1 to 240.
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. For IND560, please refer to **Appendix E**.
Label Color	[Optional] Select text color from list.
Label Status	[Integer variable, Output/Result variable, Optional] Return value of this function block. **Returned Data:** After the operator makes a 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 tx0150. 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.

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 50 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.

Properties

Index [Integer] Index of the display object in the application display. Legal values are 1 to 49. Object 50 is reserved for Single-Line Data Entry Line Prompt.

Note: The IND560 supports only 20 display objects. Indexes 1 to 19 are valid. Index 20 is reserved for the Data Entry Line.

X-Coordinate [Integer] X-Coordinate is the position of the image display relative to the top, left-hand corner of the application display window. Legal values are 1 to 320.

Y-Coordinate [Integer] Y-Coordinate is the position of the image display relative to the top, left-hand corner of the application display window. Legal values are 1 to 240.

Local Image [Optional] Image file name to display. File must be available in Local Machine.

File Name [String] Name of the file on the IND780 Compact Flash where the bitmap of the graphic display image resides.

LocalFocusedFilename [Optional] Image file name to display when the image is focused. File must be available on local PC.

Focused Filename [String, Optional] This is an optional parameter that identifies the name of the file on the Compact Flash where the bitmap of the "focused" graphic display image resides. When the user identifies this optional parameter, the TaskExpert Interpreter displays this graphical image when this object is focused on the display, and displays the File Name image when the object is not focused.

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

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.

Properties

Index [Integer] Index of the display object in the application display. Legal values are 1 to 49. Object 50 is reserved for Single-Line Data Entry Line Prompt.

Note: The IND560 supports only 20 display objects. Indexes 1 to 19 are valid. Index 20 is reserved for the Data Entry Line.

X-Coordinate [Integer] X-Coordinate is the position of the text display relative to the top, left-hand corner of the application display window. Legal values are 1 to 320.

Y-Coordinate [Integer] Y-Coordinate is the position of the text display relative to the top, left-hand corner of the application display window. Legal values are 1 to 240.

SD Name Six-character name of the Shared Data field.

Font [Optional] Font type and size of the display.

Select font from list: Arial (default) and Courier New in 10, 12, 14 and 16 point sizes.

The application can change the size of individual font using the SetFont function.

For IND560, please refer to **Appendix E**.

Color [Optional] Select text color.

Options: Black (default), Red, Blue, Green, White, Orange, Yellow, Purple.

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

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 (IND560) or 50 (IND780) 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.

Refer to the example provided in **ImageSD.zip**.

Properties:

Index [Integer] The index of the display object in the application display. Legal values are 1 to 49. Object 50 is reserved for the Data Entry Line.

Note: The IND560 supports only 20 display objects – indexes 1-19 are valid. 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. Legal values are 1 to 320.

Y-Coordinate [Integer] The position of the image display relative to the top left-hand corner of the application display window. Legal values are 1 to 240.

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

Label

Display Objects Label

Displays a label on the screen in the application window. The application window appears immediately below the Weight/SmartTrac display and above the SoftKey display.

Properties

Index	[Integer] Index of the display object in the application display. Legal values are 1 to 49. **Note**: The IND560 supports only 20 display objects. Indexes 1 to 19 are valid. Index 20 is reserved for the Data Entry Line.
X-Coordinate	[Integer] X-Coordinate is position of the text display relative to the top, left-hand corner of the application display window. Legal values are 1 to 320.
Y-Coordinate	[Integer] Y-Coordinate is position of the text display relative to the top, left-hand corner of the application display window. Legal values are 1 to 240.
Text	[Optional] Text to display.
Text ID	[Integer, Optional] Text ID used to fetch text from language database available in the Terminal.
Font	[Optional] Select the desired Font from list. For IND560, please refer to **Appendix E**.
Color	[Optional] Select text color from list.
Status	[Integer variable, Output/Result variable, Optional] Return value of this function block.

Popup Display

Display Objects Popup Display

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. Figure 13-1 shows a collapsed Popup Display object, and a Popup function block.

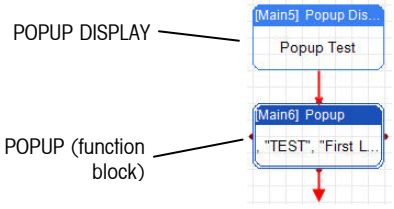


Figure 13-1: Collapsed Popup Display and Popup Function

The Popup Display can be expanded (Figure 13-2) by double clicking on the collapsed block.

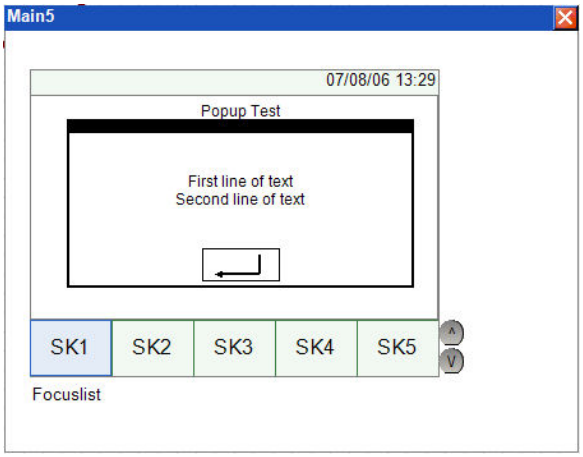


Figure 13-2: Expanded Popup Display

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

Runtime Display

Display Objects — Runtime Display

Runtime Display is used to create customized application displays. This object can be used to set the following:

IND560	IND780
Size of weight display	
	Number of weight displays
SmartTrac size	
	SmartTrac type
Softkey functions and appearance	
Contents and layout of display area	

This WYSIWYG window can be used to position any of the eight types of allowed objects onto the development screen. Up to 19 (IND560) or 49 (IND780) display objects can be placed in one screen.

Note: The highest display object number (20 for the IND560, 50 for the IND780) is reserved for the data entry line.

Refer to the appropriate sections of this document for details on the properties of these objects: Combobox, Combobox & Label, Draw Image, Draw Shared Variable, Image SD, Label, Textbox, and Textbox & Label.

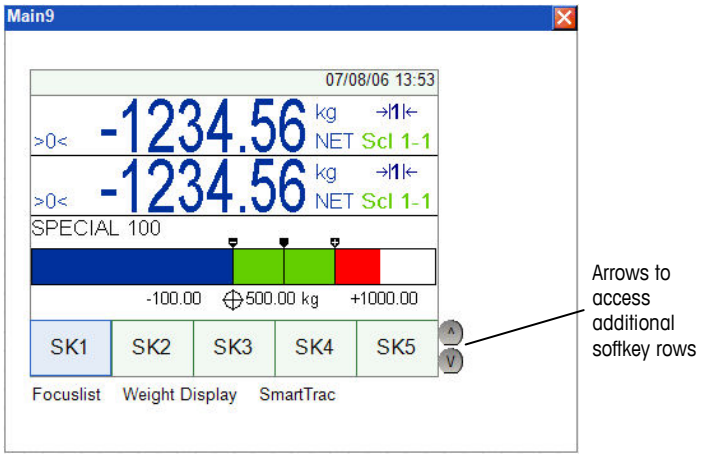


Figure 13-3: Expanded Runtime Display with Objects, IND780

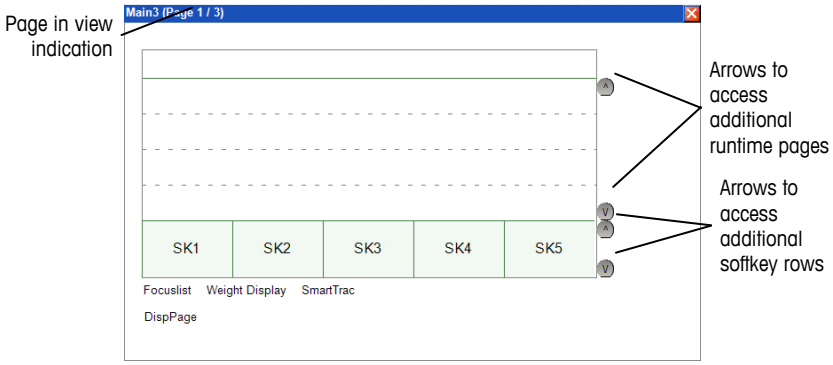


Figure 13-4: Expanded Runtime Display, Initial View, IND560

Note: Because the IND560 runtime display area is smaller, up to three page displays of the application display area can be configured. (The application display area is below the weight display and SmartTrac display, if enabled, and above the softkey row.) To select a page to configure, use the arrows indicated in Figure 13-4. Note that the page currently in view is indicated in the bar above the object. The number of pages must be set using the DispPage control – refer to page 13-14.

Once the Runtime Display function block is expanded, the developer can perform any combination of the following actions:

Note: In the Properties lists below, an asterisk (*) indicates that the item is available for IND780 only.

Control the Weight Display. Click on the Weight Display label at the bottom of the expanded display to view weight display parameters in the property window:

Properties:

Display Type* Selection of Color or Mono - used for visualization purposes only.

Scale Count* Number (1 – 5) of scales - used in conjunction with Scale property to determine how many scales to show in visualization.

Display Selection of On or Off - enables or disables the weight display.

Scale* Selection of All or Active – all enables a weight display for each scale present in the terminal; active enables a single weight display no matter how many are present

Tare* Selection of Tare, Always, Never, or Rate – enables this specific use of the tare display section of the weight display. "Tare" enables a tare display whenever there is a tare value present. "Always" enables the tare display whether there is a tare or not. "Never" disables the tare display. "Rate" enables the tare display area for the rate value.

Compress* Selection of Compress or Uncompress – compresses or uncompresses the weight display.

Size Scale Selection of Small, Medium, MediumHalf, Large, LargeHalf, Larger, or Huge (IND780) and Small, Medium or Large (IND560) – sets the size of the weight display to use. In the IND780, half size selections place two weight displays side-by-side across the screen.

Size Sum* Selection of Sum Small, Sum Medium, Sum MediumHalf, Sum Large, Sum LargeHalf, Sum Larger, or Sum Huge – picks the size of the sum weight display to use. Half size selections place two weight displays side-by-side across the screen.

Control the SmartTrac Display. Click on the SmartTrac label at the bottom of the expanded display to view SmartTrac display parameters in the property window:

Properties:

Display Type* Selection of Color or Mono - used for visualization purposes only.

Setpoint Number of active target used for operation.

Display Selection of On or Off - enables or disables the weight display.

Size Selection of Small, Medium or Large – picks the size of the SmartTrac display.

SmartTrac Type* Selection of Bar graph, Cross Hairs or Three Zone – picks the type of SmartTrac display.

> PROGRAMMING TIP: When creating multiple setup pages, only the first page requires the Weight Display and SmartTrac to be turned OFF if either is already ON during runtime. Leaving these two parameters blank on sub-sequent pages will allow each page to be drawn much faster on the display.

Control the Focuslist. Click on the Focuslist label at the bottom of the expanded display to view display focus list control parameters in the property window. Double-click the Focuslist label to show each display object's Focuslist order next to it on the display.

Note: The numbers that appear are **not** the indexes for the objects, but their order in the Focuslist.

Properties:

Object Index(s)	List of indexes of display objects to pass the focus.
Status	Optional output/result variable.

Control the Softkeys. Click on any of the softkey locations to view its control parameters in the property window. Each position has a fixed location. SK1 – 5 are shown by default, but SK6 – 10 and SK11 – 15 can be selected by using the up/down arrows.

Properties:

Display Type*	Selection of Color or Mono - used for visualization purposes only.
Local Image	Local file path and filename – used for visualization purposes within TaskExpert only (i.e., emulates the on-screen appearance of the softkey).
Softkey ID	Number related to softkey position (used by terminal to determine placement) – this value is read only and is determined by the tool based on location of selected softkey.
Mode	An integer variable or the selection of Delete, Insert, Move Down or Overwrite – this variable or direct selection is used to determine what kind of softkey operation is desired. Legal values for the variable are **overwrite existing**, **insert & move down existing**, and **delete**.
Key ID	An integer variable or value that is passed from the Softkey manager when the softkey is pressed. This value is used by applications to identify which softkey has been pressed.
Display Item	A string variable or "text" that contains either the specific text or a softkey image path and name that the terminal will use to represent the softkey on-screen.
Executable File	Optional string variable or "text" that contains the path and name of the application that should be run when the softkey is pressed.
Status	Optional output/result variable.

Select the Softkey command type. Click on the empty space inside the runtime frame to view softkey command type parameters in the property window.

Properties:

Name	Optional display name (used for commenting and documentation).
Softkey*	Selection of Replace, Push, or Push Persist – chooses which type of softkey command is used for all softkeys configured by this runtime display.

Set the Display Page (IND560 only). Click on the DispPage label at the bottom of the expanded display to view Display Page control parameter in the property

window. As noted, the IND560 permits multiple-page displays of application data.

Properties:

Page Set the number of display pages – 1, 2 or 3. This indicates to the IND560 operating system how many pages have been configured. Note that, even if multiple pages have been created and are accessible using the up and down arrows (see Figure 13-4), only the number of pages set here will be available in the terminal.

Figure 13-5 show runtime displays for the IND780 and IND560, respectively, with various objects visible.

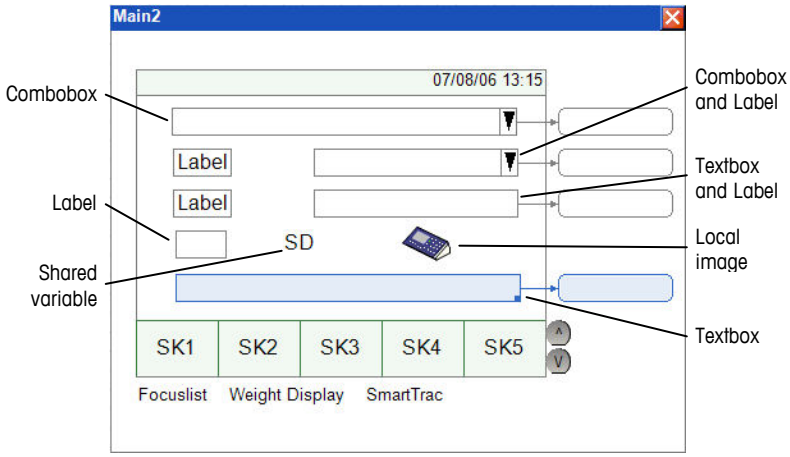


Figure 13-5: Runtime Display, IND780

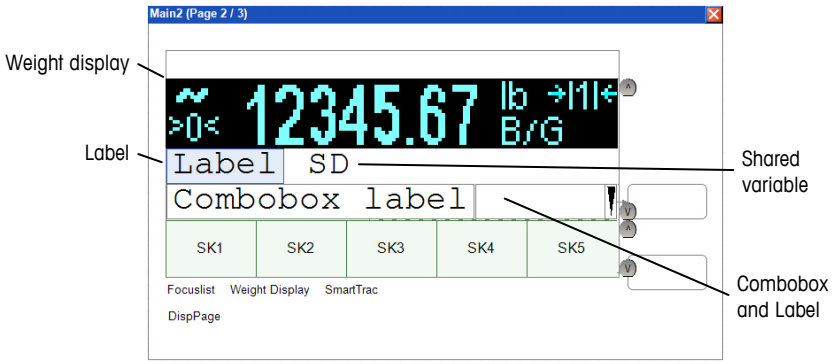


Figure 13-6: Runtime Display, IND560

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.

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.

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.

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.

Properties

Index	[Integer] Index of the display object in the application display. Legal values are 1 to 49. Object 50 is reserved for Single-Line Data Entry Line Prompt. **Note**: The IND560 supports only 20 display objects. Indexes 1 to 19 are valid. Index 20 is reserved for the Data Entry Line.
Index Variable Instance	[Integer, Optional] Index of the shared variable to be associated. Legal values are 1 to 49.
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. Legal values are 1 to 320.

Y-Coordinate Position of the text display relative to the top, left-hand corner of the application display window. Legal values are 1 to 240.

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 numeric data entry mode, %nn is the format specification where nn is the max number of numeric digits. The entered numeric data is left aligned. The TEXTBOX accepts only numeric values, and it ignores alpha keys. The operator must key in the decimal point if required. 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, the focus returns to the control.

In password entry mode, "*ss" is the format specification where *ss* is the maximum number of alphanumeric characters. Data entered into the Textbox appears in left-to-right order, with each character getting replaced by an asterisk (*) as the user keys in the information.

Font [Optional] Font type and size of the display.

Select font from list: Arial (default) and Courier New in 10, 12, 14 and 16 point sizes.

The application can change the size of individual font using the SetFont function.

IND560 font options are Alpha, Chinese, Num1 and Num5. Please refer to **Appendix E**.

Color [Optional] Select text color.

Options: Black (default), Red, Blue, Green, White, Orange, Yellow, Purple.

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 tx0150. 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.

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.

Properties

Index	[Integer] Index of the display object in the application display. Legal values are 1 to 49. Object 50 is reserved for Single-Line Data Entry Line Prompt. **Note**: The IND560 supports only 20 display objects. Indexes 1 to 19 are valid. Index 20 is reserved for the Data Entry Line.
Index Variable Instance	[Integer, Optional] Index of the shared variable to be associated. Legal values are 1 to 49.
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. Legal values are 1 to 320.

Y-Coordinate Position of the text display relative to the top, left-hand corner of the application display window. Legal values are 1 to 240.

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 numeric data entry mode, %nn is the format specification where nn is the max number of numeric digits. The entered numeric data is left aligned. The TEXTBOX accepts only numeric values, and it ignores alpha keys. The operator must key in the decimal point if required. 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, the focus returns to the control.

Font [Optional] Font type and size of the display.

Select font from list: Arial (default) and Courier New in 10, 12, 14 and 16 point sizes.

The application can change the size of individual font using the SetFont function.

IND560 font options are Alpha, Chinese, Num1 and Num5. Please refer to **Appendix E**.

Color [Optional] Select text color.

Options: Black (default), Red, Blue, Green, White, Orange, Yellow, Purple.

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 49.

Label X-Coordinate	[Integer] X-Coordinate is position of the text display relative to the top, left-hand corner of the application display window. Legal values are 1 to 320.
Label Y-Coordinate	[Integer] Y-Coordinate is position of the text display relative to the top, left-hand corner of the application display window. Legal values are 1 to 240.
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 tx0150. 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.

Chapter 14 Event and Error Handling Commands

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.
```

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 remote serial port, only one can have the port open at a time.	Correct TaskExpert application. To share remote 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.	
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.	

Error Code	Error Message	Description	Probable Cause	Remedy	Trapped with ON ERROR*
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 directory of the Storage Card (IND780) or Flash2:\ memory (IND560). 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.	IND780: The program exceeds 10000 text lines or 400 KB. IND560: The program exceeds 5000 text lines or 100 KB.	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 (IND780)* or 1000 (IND560) characters. *Version 6.1.xx firmware (and higher) length is changed to 500 due to UNICODE support	Correct TaskExpert program.	

Error Code	Error Message	Description	Probable Cause	Remedy	Trapped with ON ERROR*
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 listed above.	Correct TaskExpert program.	•
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 in the IND780 terminal cluster.	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.	

Error Code	Error Message	Description	Probable Cause	Remedy	Trapped with ON ERROR*
42	Baseboard Switch 2 Set	SW-2 (IND780) is set ON. SW2-1 (IND560) 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.

Clear Event

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

Refer to the example provided in **EventHandling.zip**.

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.

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.

Delete Event

De-allocates an event.

Refer to the example provided in **EventControl.zip**.

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.

Disable

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

Refer to the example provided in **EventControl.zip**.

Enable

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

Refer to the example provided in **EventControl.zip**.

Error

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

Refer to the example provided in **ErrorHandling.zip**.

Properties:

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

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.

Refer to the example provided in **ErrorHandling.zip**.

Properties:

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

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.

Refer to the example provided in **ErrorHandling.zip**.

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.

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.

Refer to the example provided in **EventHandling.zip**.

Properties:

Event Select keyboard or timer event.

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.

Refer to the example provided in **ErrorHandling.zip**.

Properties:

Call Type Type of ON ERROR call.
Options: GOTO, Subroutine

Block/Function Select the error handling block or error handling subroutine.

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.

Refer to the example provided in **EventHandling.zip**.

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.

WaitEvent

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

Refer to the example provided in **EventHandling.zip**.

Chapter 15 Ladder Commands

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.

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.

For examples of ACC usage and of the commands described in this document, refer to **Ladder.zip**.

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 15-1.

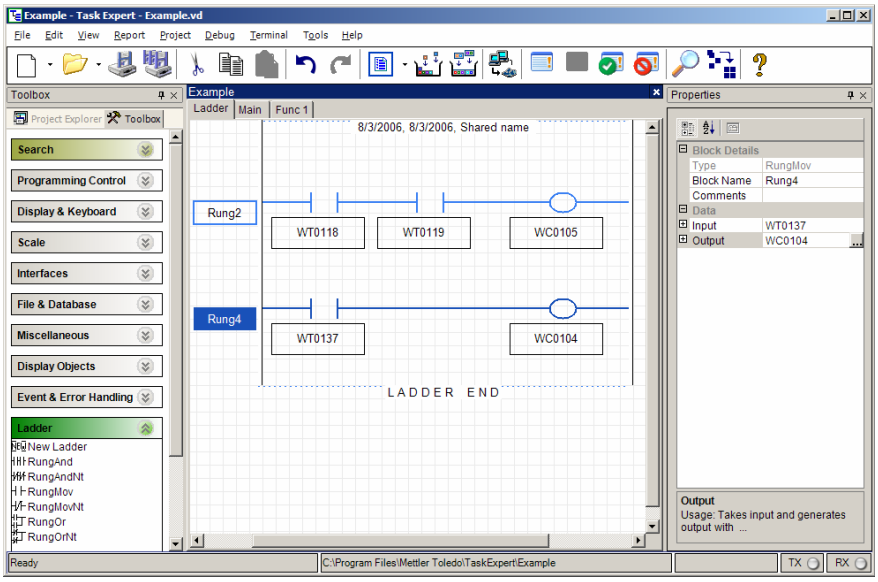


Figure 15-1: Ladder Routine, Shared Name Display Type

Setting the "Display Type" property to "Alias name" would display all the shared variables in the ladder in alias name as shown in Figure 15-2.

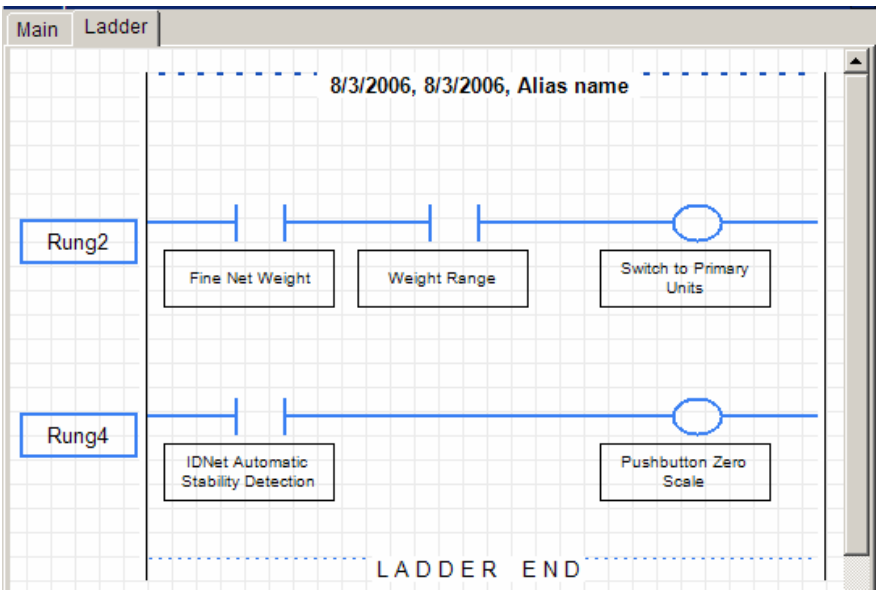


Figure 15-2: Ladder Routine, Alias Name Display Type

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.

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.

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.

All the example **.zip** files contain examples of the Notes block.

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 1 feeding" to generate a physical discrete output.

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.

RungMov

Takes an input and generates an output with the same value. For example, take a tare on Scale 2 when a physical discrete input goes on.

RungMovNt

Moves the inverse of the input to the output. For example, turn on a physical discrete output when the data from Scale 1 is invalid.

RungOr

Takes two inputs, OR's them together, and outputs the result. For example, turn on a physical discrete output if Scale 1 or Scale 2 is in motion.

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.

Ladder Logic Interpreter

The IND780 and IND560 have 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 Scale 2.
- A Task Expert application program must build simple Ladder Logic programs in order to customize the monitoring of the Discrete I/O.

Appendix A Building a Custom Library

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.

TaskExpert Capabilities

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.

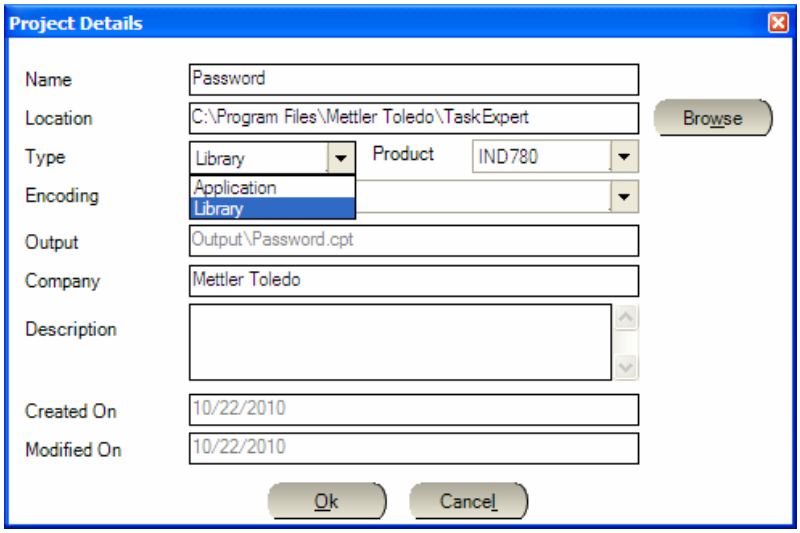


Figure A-1: Selecting Library as Project Type

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.

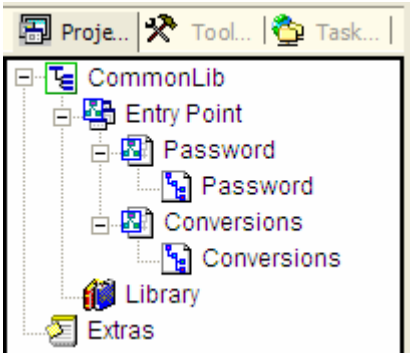


Figure A-2: Entry Points Displayed in Project Explorer

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.

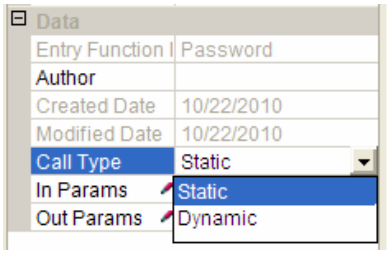


Figure A-3: Selecting Type of Library

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.

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.

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.

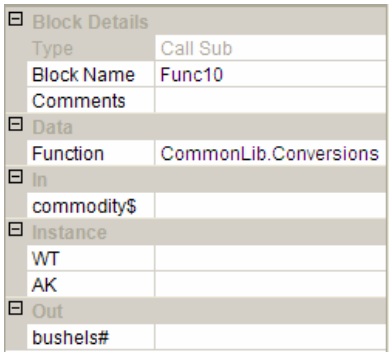


Figure A-4: Shared Data Instances

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.

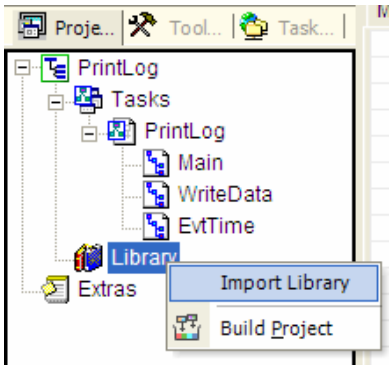
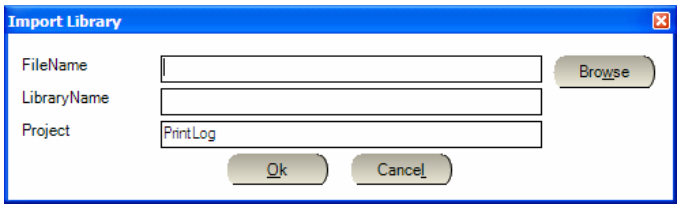


Figure A-5: Importing a Library

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.



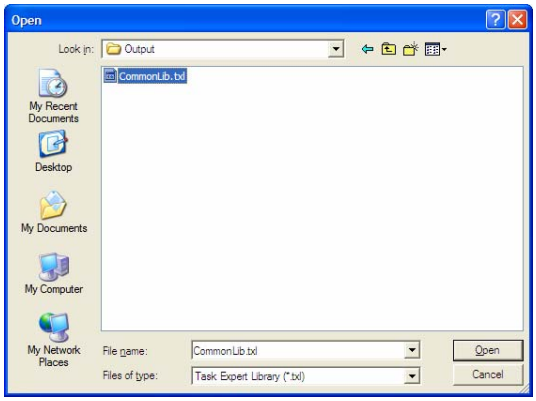


Figure A-6: Finding a Library to Import

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.

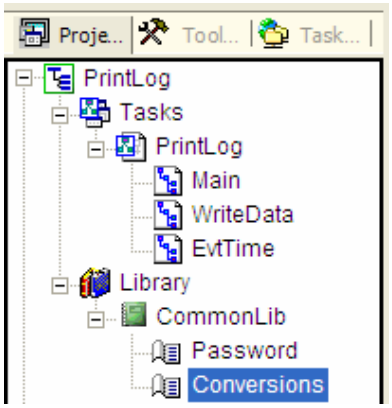


Figure A-7: Library and Associated Functions Displayed in Project Explorer

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.

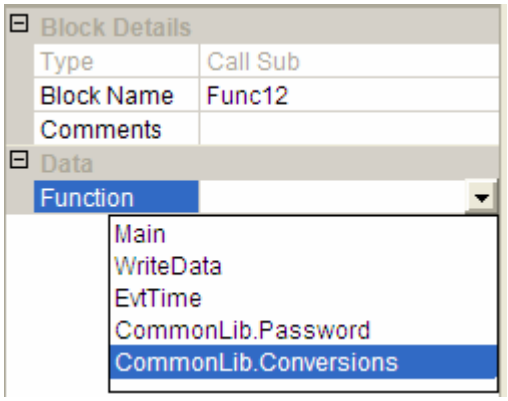


Figure A-8: Library Functions in Function Block Drop-Down Menu

Appendix B SQL CE Command Reference

Create Index

Creates an index on a given table.

Refer to the example provided in **CreateIndex.zip**.

Syntax

```
CREATE [ UNIQUE ] INDEX index_name
   ON table ( column [ ASC | DESC ] [ ,...n ] )
```

Arguments

UNIQUE

Creates a unique index (one in which no two rows are permitted to have the same index value) on a table.

Microsoft® SQL Server™ 2000 Windows® CE Edition (SQL Server CE) checks for duplicate values when the index is created (if data already exists) and checks each time data is added with an INSERT or UPDATE statement. Duplicates must be eliminated before a unique index can be created on the column(s). If duplicate key values exist, the CREATE INDEX statement is canceled and an error is returned. A unique index can be created only on columns that are defined as NOT NULL.

When a unique index exists, UPDATE or INSERT statements that would generate duplicate key values are rolled back, and SQL Server CE returns an error. This is true even if the UPDATE or INSERT statement changes many rows but causes only one duplicate.

index_name

Is the name of the index. Index names must be unique within a table but do not need to be unique within a database. Index names must follow the rules of identifiers.

table

Is the table that contains the column or columns to be indexed.

column

Is the column or columns to which the index applies. Specify two or more column names to create a composite index on the combined values in the specified columns. List the columns to be included in the composite index (in sort-priority order) inside the parentheses after *table*.

Note: Columns consisting of the ntext or image data types cannot be specified as columns for an index.

Important: All columns in a UNIQUE INDEX must have NOT NULL specified.

[ASC | DESC]

Determines the sort direction for the particular index column: ASC, the default, is ascending; and DESC is descending.

n

A placeholder indicating that multiple columns can be specified for any particular index. The maximum number of columns that can participate in an index is 10.

Examples

The following example creates a unique index on the MyCustomers table.

```
CREATE TABLE MyCustomers (CustID int, CompanyName
nvarchar(50))
CREATE UNIQUE INDEX idxCustId ON MyCustomers (CustId)
```

Create Table

Creates a new table.

Refer to the example provided in CustomDatabase.zip.

Syntax

```
CREATE TABLE table_name
   ( { < column_definition > | < table_constraint > }
[ ,...n ]
   )
< column_definition > ::=
   { column_name data_type }
   [ { DEFAULT constant_expression
      | [ IDENTITY [ ( seed , increment ) ]
      ]
    } ]
   [ < column_constraint > [ ...n ] ]
< column_constraint > ::=
   [ CONSTRAINT constraint_name ]
   { [ NULL | NOT NULL ]
      | [ PRIMARY KEY | UNIQUE ]
      | REFERENCES ref_table [ ( ref_column ) ]
      [ ON DELETE { CASCADE | NO ACTION } ]
      [ ON UPDATE { CASCADE | NO ACTION } ]
   }
< table_constraint > ::=
   [ CONSTRAINT constraint_name ]
   { [ { PRIMARY KEY | UNIQUE }
      { ( column [ ,...n ] ) }
      ]
   | FOREIGN KEY
      [ ( column [ ,...n ] ) ]
```

```
REFERENCES ref_table  ( ref_column [ ,...n ] )
   [ ON DELETE { CASCADE | NO ACTION } ]
   [ ON UPDATE { CASCADE | NO ACTION } ]
}
```

Arguments

table_name

Is the name of the new table. Table names must conform to the rules for identifiers. *table_name* must be unique within the database. *table_name* can contain a maximum of 128 characters.

column_name

Is the name of a column in the table. Column names must conform to the rules for identifiers and must be unique in the table.

data_type

Specifies the column data type. For information about data types, see Data Types.

DEFAULT

Specifies the value provided for the column when a value is not explicitly supplied during an insert action. DEFAULT definitions can be applied to any column, except those defined by the IDENTITY property. DEFAULT definitions are removed when the table is dropped. A constant value, such as a character string, or a date function can be used as a default.

IDENTITY

Indicates that the new column is an identity column. When a new row is added to the table, Microsoft® SQL Server™ 2000 Windows® CE Edition (SQL Server CE) provides a unique, incremental value for the column. Identity columns are commonly used in conjunction with PRIMARY KEY constraints to serve as the unique row identifier for the table. The IDENTITY property can be assigned only to **int** columns. Only one identity column can be created per table. Bound defaults and DEFAULT constraints cannot be used with an identity column. You must specify both the seed and increment or neither. If neither is specified, the default is (1,1).

seed

Is the value used for the first row that is loaded into the table.

increment

Is the incremental value added to the identity value of the previous row that is loaded.

CONSTRAINT

Is an optional keyword indicating the beginning of a PRIMARY KEY, UNIQUE, or FOREIGN KEY constraint definition. Constraints are special properties that enforce data integrity and create special types of indexes for the table and its columns.

constraint_name

Is the name of a constraint. *constraint_name* is optional and must be unique within a database. If *constraint_name* is not specified, SQL Server CE generates a constraint name.

NULL | NOT NULL

Are keywords that specify whether null values are allowed in the column. NULL is not strictly a constraint but can be specified in the same manner as NOT NULL.

PRIMARY KEY

Is a constraint that enforces entity integrity for a given column or columns using a unique index. Only one PRIMARY KEY constraint can be created per table.

UNIQUE

Is a constraint that provides entity integrity for a given column or columns using a unique index. Columns in a UNIQUE constraint must also be NOT NULL. A table can have multiple UNIQUE constraints.

Note: SQL Server CE can use indexes to enforce PRIMARY KEY and UNIQUE constraints. You should not rely on this behavior nor try to manipulate any indexes that are created as part of a constraint.

FOREIGN KEY...REFERENCES

Is a constraint that provides referential integrity for the data in the column. FOREIGN KEY constraints require that each value in the column exists in the specified column in the referenced table.

ref_table

Is the name of the table referenced by the FOREIGN KEY constraint.

(*ref_column* [,...*n*])

Is a column, or list of columns, from the table referenced by the FOREIGN KEY constraint.

ON DELETE {CASCADE | NO ACTION}

Specifies what action takes place to a row in the table that is created when that row has a referential relationship and the referenced row is deleted from the parent table. The default is NO ACTION.

If CASCADE is specified, a row is deleted from the referencing table when the corresponding referenced row is deleted from the parent table. If NO ACTION is specified, SQL Server CE raises an error and the delete action on the referenced row in the parent table is rolled back.

For example, in the **Northwind** database, the **Orders** table has a referential relationship with the **Customers** table: the **Orders.CustomerID** foreign key references the **Customers.CustomerID** primary key.

If a DELETE statement is executed on a row in the **Customers** table and an ON DELETE CASCADE action is specified for **Orders.CustomerID**, SQL Server CE checks for one or more dependent rows in the **Orders** table. If any exists, the dependent rows in the **Orders** table are deleted, as well as the row referenced in the **Customers** table.

Alternately, if NO ACTION is specified, SQL Server CE raises an error and rolls back the delete action on the row in the **Customers** table when there is at least one row in the **Orders** table that references it.

ON UPDATE {CASCADE | NO ACTION}

Specifies what action takes place to a row in the table that is created when that row has a referential relationship and the referenced row is updated in the parent table. The default is NO ACTION.

If CASCADE is specified, the row is updated in the referencing table when the corresponding referenced row is updated in the parent table. If NO ACTION is specified, SQL Server CE raises an error and the update action on the referenced row in the parent table is rolled back.

For example, in the **Northwind** database, the **Orders** table has a referential relationship with the **Customers** table: the **Orders.CustomerID** foreign key references the **Customers.CustomerID** primary key.

If an UPDATE statement is executed on a row in the **Customers** table and an ON UPDATE CASCADE action is specified for **Orders.CustomerID**, SQL Server CE checks for one or more dependent rows in the **Orders** table. If any exists, the dependent rows in the **Orders** table are updated, as well as the row referenced in the **Customers** table.

Alternately, if NO ACTION is specified, SQL Server CE raises an error and rolls back the update action on the referenced row in the **Customers** table when there is at least one row in the **Orders** table that references it.

column

Is a column or list of columns, in parentheses, used in table constraints to indicate the columns used in the constraint definition.

Remarks

Constraints

PRIMARY KEY Constraints

- A table can contain only one PRIMARY KEY constraint.
- Each PRIMARY KEY generates an index.
- All columns defined within a PRIMARY KEY constraint must be defined as NOT NULL. If nullability is not specified, all columns participating in a PRIMARY KEY constraint have their nullability set to NOT NULL.

UNIQUE Constraints

- Each UNIQUE constraint generates an index.
- All columns defined as a UNIQUE constraint must be defined as NOT NULL. If nullability is not specified, all columns defined as UNIQUE constraints have their nullability set to NOT NULL.

FOREIGN KEY Constraints

- When a value other than NULL is entered into the column of a FOREIGN KEY constraint, the value must exist in the referenced column; otherwise, a foreign key violation error message is returned.

- FOREIGN KEY constraints can reference another column in the same table (a self-reference). However, FOREIGN KEY constraints cannot be used to create a self-referencing or circular FOREIGN KEY constraint.
- The REFERENCES clause of a column-level FOREIGN KEY constraint can list only one reference column, which must have the same data type as the column on which the constraint is defined.
- The REFERENCES clause of a table-level FOREIGN KEY constraint must have the same number of reference columns as the number of columns in the constraint column list. The data type of each reference column also must be the same as the corresponding column in the column list.
- FOREIGN KEY constraints can reference only columns in PRIMARY KEY or UNIQUE constraints in the referenced table. FOREIGN KEY constraints cannot reference unique indexes.

Additional Constraint Information

- An index created for a constraint cannot be dropped with the DROP INDEX statement; the constraint must be dropped with the ALTER TABLE DROP CONSTRAINT statement.
- Constraint names must follow the rules for identifiers, except that the name cannot begin with a number sign (#). If the CONSTRAINT keyword and *constraint_name* is not supplied, a system-generated name is assigned to the constraint.
- When a constraint is violated in an INSERT, UPDATE, or DELETE statement, the statement is terminated.

DEFAULT Definitions

A column can have only one DEFAULT definition, which can contain constant values or constant functions.

Nullability Rules Within a Table Definition

The nullability of a column determines whether or not that column can allow a null value (NULL) as the data in that column. NULL is not zero or blank: it means no entry was made or an explicit NULL was supplied, and it usually implies that the value is either unknown or not applicable.

Examples

The following example creates a two-column table with an identity column as the PRIMARY KEY.

```
CREATE TABLE MyCustomers (CustID int IDENTITY (100,1)
PRIMARY KEY, CompanyName nvarchar (50))
```

The following example creates a one-column table with a PRIMARY KEY constraint.

```
CREATE TABLE MyCustomers2 (CustID int CONSTRAINT
pkCustId PRIMARY KEY)
```

The following example creates a table with one of its columns referencing a column in another table.

```
CREATE TABLE MyOrders (OrderID int, CustID int
REFERENCES MyCustomers(CustID))
```

Data Types

Each column in a table in Microsoft® SQL Server™ 2000 Windows® CE Edition (SQL Server CE) has a data type that specifies the type of data (such as **integer**, **character**, or **money**) that the column can hold.

Note: There may be minor differences between Microsoft SQL Server 2000 and SQL Server CE in the way data types are promoted when the execution of a function results in an overflow or underflow. You might have to perform explicit CAST in SQL Server 2000 to get the same behavior in SQL Server CE.

SQL Server CE supports the following data types.

Data type	Description
integer	Integer (whole number) data from -2^31 (-2,147,483,648) through 2^31 - 1 (2,147,483,647).
float	Floating point number data from -1.79E + 308 through 1.79E + 308 Storage size is 8 bytes.
national character(*n*) **Synonym:** nchar(*n*)	Fixed-length Unicode data with a maximum length of 255 characters. Default length = 1 Storage size, in bytes, is two times the number of characters entered.
national character varying(*n*) **Synonym:** nvarchar(*n*)	Variable-length Unicode data with a length of 1 to 255 characters. Default length = 1 Storage size, in bytes, is two times the number of characters entered.
IDENTITY [(*s*, *i*)]	This is a property of a data column, not a distinct data type. Only data columns of the integer data types can be used for identity columns. A table can have only one identity column. A seed and increment can be specified and the column cannot be updated. *s* (seed) = starting value *i* (increment) = increment value

Delete

Removes rows from a table.

Refer to the example provided in **Delete.zip**.

Syntax

```
DELETE
   [ FROM ] table_name
   [ WHERE < search_condition > ]
```

Arguments

FROM

Is an optional keyword that can be used between the DELETE keyword and the target *table_name*.

table_name

Is the name of the table from which the rows are to be removed.

WHERE

Specifies the conditions used to limit the number of rows that are deleted.

<search_condition>

Specifies the restricting conditions for the rows to be deleted. There is no limit to the number of predicates that can be included in a search condition.

Remarks

If a WHERE clause is not supplied, DELETE removes all the rows from the table.

If a search condition is specified, it is applied to each row of the table; and all rows for which the result of the search condition is TRUE are marked for deletion.

The search condition is evaluated for each row of the table before any deletions occur.

All rows that are marked for deletion are deleted at the end of the DELETE statement prior to the checking of any integrity constraint.

The DELETE statement might fail if it violates a FOREIGN KEY constraint. If the DELETE removes multiple rows and any one of the removed rows violates a constraint, the statement is canceled, an error is returned, and no rows are removed.

Examples

A. Using DELETE with no parameters

The following example deletes all rows from the **Customers** table in the **Northwind** database.

```
DELETE Customers
```

B. Using DELETE on a set of rows

The following example uses the **Customers** table in the **Northwind** database. Because **CompanyName** may not be unique, the following example deletes all rows in which **CompanyName** is Eastern Connection.

```
DELETE FROM Customers WHERE CompanyName = 'Eastern
Connection'
```

DROP INDEX

Removes an index from the current database.

Refer to the example provided in **CreateIndex.zip**.

The DROP INDEX statement does not apply to indexes created by defining PRIMARY KEY or UNIQUE constraints (created by using the PRIMARY KEY or UNIQUE options of either the CREATE TABLE or ALTER TABLE statements). For more information about PRIMARY KEY or UNIQUE constraints, see CREATE TABLE.

Syntax

```
DROP INDEX 'table_name.index_name'
```

Arguments

table_name

Is the name of the table in which the indexed column is located. Table names must conform to the rules for identifiers.

index_name

Is the name of the index to be dropped. Index names must conform to the rules for identifiers.

Remarks

After DROP INDEX is executed, all the space previously occupied by the index is regained. This space then can be used for any database object.

DROP INDEX cannot be specified on an index on a system table.

DROP TABLE

Removes a table definition and all data, indexes, constraints, and permission specifications for that table.

Refer to the example provided in **DropTable.zip**.

Syntax

```
DROP TABLE table_name
```

Arguments

table_name

Is the name of the table to be removed.

Remarks

DROP TABLE cannot be used to drop a table referenced by a FOREIGN KEY constraint. The referencing FOREIGN KEY constraint or the referencing table must be dropped first.

When a table is dropped, rules or defaults on it lose their binding, and any constraints associated with it are automatically dropped. If you re-create a table, you must rebind the appropriate rules and defaults, and add all necessary constraints.

You cannot use the DROP TABLE statement on system tables.

EXISTS

Specifies a subquery to test for the existence of rows.

Refer to the example provided in **Exists.zip**.

Syntax

```
EXISTS subquery
```

Arguments

subquery

Is a restricted SELECT statement.

Result Types

bit

Result Values

Returns TRUE if a subquery contains any rows.

Examples

The following example finds, from the **Orders** table, all the orders with Washington as the Shipping Region for Employees who are listed in the **Employees** table in the **Northwind** database.

```
SELECT * FROM Orders WHERE ShipRegion = 'WA' AND
EXISTS (SELECT EmployeeID FROM Employees AS Emp WHERE
Emp.EmployeeID = Orders.EmployeeID)
```

Expressions

An expression is a combination of symbols and operators that the database system evaluates to obtain a single data value. Simple expressions can be a single constant, variable, column, or scalar function. Operators can be used to join two or more simple expressions into a complex expression.

Syntax

```
{ constant
   | scalar_function
   | [ alias. ] column
   | ( expression )
   | { unary_operator } expression
   | expression { binary_operator } expression
}
```

Arguments

constant

Is a symbol that represents a single, specific data value. *constant* is one or more alphanumeric characters (letters a-z and A-Z) or symbols (such as !, @, #). Unicode character and **datetime** values are enclosed in quotation marks; binary strings and numeric constants are not.

scalar_function

Is a unit of SQL syntax that provides a specific service and returns a single value.

[alias.]

Is the alias, or correlation name, assigned to a table by the AS keyword in the FROM clause.

column

Is the name of a column.

(expression)

Is any valid expression in Microsoft® SQL Server™ 2000 Windows® CE Edition (SQL Server CE) as defined in this topic. The parentheses are grouping operators that ensure that all the operators in the expression within the parentheses are evaluated before the resulting expression is combined with another.

{ unary_operator }

Is an operator that has only one numeric operand:

- + indicates a positive number.
- - indicates a negative number.
- ~ indicates the complement operator.

Unary operators can be applied only to expressions that evaluate to any of the data types of the numeric data type category.

{ binary_operator }

Is an operator that defines the way two expressions are combined to yield a single result. *binary_operator* can be an arithmetic operator, the assignment operator (=), a bitwise operator, a comparison operator, a logical operator, the string concatenation operator (+), or a unary operator. For more information, see Operators.

Expression Results

For a simple expression built of a single constant, variable, scalar function, or column name, the data type, precision, scale, and value of the expression is the data type, precision, scale, and value of the referenced element.

When two expressions are combined using comparison or logical operators, the resulting data type is Boolean and the value is one of three values: TRUE, FALSE, or UNKNOWN.

When two expressions are combined using arithmetic, bitwise, or string operators, the resulting data type is determined by the operator.

Complex expressions made up of many symbols and operators evaluate to a single-valued result. The data type, precision, and value of the resulting expression are determined by combining the component expressions two at a time until a final result is reached. The sequence in which the expressions are combined is defined by the precedence of the operators in the expression.

Remarks

An operator can combine two expressions if they both have data types supported by the operator and at least one of the following conditions is TRUE:

- The expressions have the same data type.

- The data type with the lower precedence can be implicitly converted to the data type with the higher data type precedence.

If there is no supported implicit conversion, the two expressions cannot be combined.

In a programming language such as Microsoft Visual Basic®, an expression always evaluates to a single result. Expressions in an SQL select list have a variation on this rule: The expression is evaluated individually for each row in the result set. A single expression may have a different value in each row of the result set, but each row has only one value for the expression. For example, in this SELECT statement, both the reference to ProductID and the term 1+2 in the select list are expressions:

```
SELECT ProductID, 1+2
FROM Products
```

The expression 1+2 evaluates to 3 in each row in the result set. Although the expression ProductID generates a unique value in each result set row, each row only has one value for ProductID.

FROM Clause

Specifies the table(s) from which to retrieve rows. In Microsoft® SQL Server™ 2000 Windows® CE Edition (SQL Server CE), the FROM clause is always required; otherwise, an error is returned.

Refer to the example provided in **InQuery.zip**.

Syntax

```
FROM { < table_source > } [ ,...n ]
< table_source > ::=
   table_name [ [ AS ] table_alias ]
   | < joined_table >
< joined_table > ::=
   < table_source > < join_type > < table_source > ON
< search_condition >
   | ( < joined_table > )
< join_type > ::=
   [ INNER | { { LEFT | RIGHT } [ OUTER ] } ] JOIN
```

Arguments

< table_source >

Specifies the tables and joined tables for the SELECT statement.

table_name [[AS] table_alias]

Specifies the name of a table and an optional alias.

< joined_table >

Is a result set that is the join of two or more tables.

For multiple joins, you can use parentheses to specify the order of the joins.

< join_type >

Specifies the type of join operation.

INNER

Specifies that all matching pairs of rows are returned. Discards unmatched rows from both tables. This is the default if no join type is specified.

LEFT [OUTER]

Specifies that all rows from the left table that are not meeting the specified condition are included in the result set in addition to all rows returned by the inner join. Output columns from the left table are set to NULL.

RIGHT [OUTER]

Specifies that all rows from the right table that are not meeting the specified condition are included in the result set in addition to all rows returned by the inner join. Output columns from the right table are set to NULL.

JOIN

Indicates that the specified tables should be joined.

ON < search_condition >

Specifies the condition on which the join is based. The condition can specify any valid predicate, although columns and comparison operators are often used.

IDENTITY (Property)

Creates an identity column in a table. This property is used with the CREATE TABLE statements.

Refer to the example provided in **Identity.zip**.

Syntax

```
IDENTITY [ ( seed , increment ) ]
```

Arguments

seed

Is the value that is used for the first row loaded into the table.

increment

Is the incremental value that is added to the identity value of the previous row that was loaded.

Note: You must specify both the seed and increment, or neither. If neither is specified, the default is (1,1).

Remarks

In Microsoft® SQL Server™ 2000 Windows® CE Edition (SQL Server CE), the IDENTITY property can be created only on a column of an **integer** data type. A table can have only one IDENTITY column.

Examples

This example creates a two-column table in which the first column is an IDENTITY column.

```
CREATE TABLE MyCustomers (CustID int IDENTITY (100,1)
PRIMARY KEY, CompanyName nvarchar (50))
```

IN

Determines whether a given value matches any value in a subquery or a list.

Refer to the example provided in **InQuery.zip**.

Syntax

```
test_expression [ NOT ] IN
   (
      subquery
      | expression [ ,...n ]
   )
```

Arguments

test_expression

Is any valid expression in Microsoft® SQL Server™ 2000 Windows® CE Edition (SQL Server CE).

subquery

Is a subquery that has a result set of one column. This column must have the same data type as *test_expression*.

expression [,...n]

Is a list of expressions to test for a match. All expressions must be of the same type as *test_expression*.

Result Types

bit

Result Value

If the value of *test_expression* is equal to any value returned by *subquery* or is equal to any *expression* from the comma-separated list, the result value is TRUE. Otherwise, the result value is FALSE.

Using NOT IN negates the returned value.

Examples

The following example selects all customers in the **Northwind** database who are from Brazil, Argentina, and Venezuela.

```
SELECT * FROM Customers WHERE Country IN ('Brazil',
'Argentina', 'Venezuela')
```

INSERT

Adds new rows to a table.

Refer to the example provided in CustomDatabase.zip.

Syntax

```
INSERT [ INTO]
   table_name  [ ( column_list ) ]
      { VALUES
      ( { DEFAULT | NULL | expression } [ ,...n] )
       }
```

Arguments

[INTO]

Is an optional keyword that can be used between INSERT and the target table.

table_name

Is the name of a table that is to receive the data.

(column_list)

Is a list of one or more columns in which to insert data. *column_list* must be enclosed in parentheses and delimited by commas.

VALUES

Introduces the list of data values to be inserted. There must be one data value for each column in *column_list* (if specified) or in the table. The values list must be enclosed in parentheses.

DEFAULT

Requires that the default value defined for a column is to be used by Microsoft® SQL Server™ 2000 Windows® CE Edition (SQL Server CE).

NULL

Indicates that the value is unknown. A value of NULL is different from an empty or zero value.

expression

Is a constant, a variable, or an expression.

Remarks

To replace data in a table, the DELETE statement must be used to clear existing data before loading new data with INSERT. To modify column values in existing rows, use UPDATE.

If the insert *column_list* is omitted, then an insert column list that identifies all columns of the table in the ascending sequence of their ordinal positions is implicit.

A column in the table can be identified only once in *column_list.*

If a column is not in *column_list,* SQL Server CE must be able to provide a value based on the definition of the column; otherwise, the row cannot be loaded. SQL Server CE automatically provides a value for the column if the column:

- Has an IDENTITY property. The next incremental identity value is used.
- Has a default. The default value for the column is used.
- Is nullable. A null value is used.

The column list and VALUES list must be used when inserting explicit values into an identity column. If the values in the VALUES list are not in the same order as the columns in the table or do not have a value for each column in the table, *column_list* must be used to explicitly specify the column that stores each incoming value.

When DEFAULT is used to specify a column value, the default value for that column is inserted. If a default does not exist for the column and the column allows null values, NULL is inserted. DEFAULT is not valid for an identity column.

When you insert rows, the following rules apply:

- If a value is being loaded into columns with an **nchar** or **nvarchar**, the padding or truncation of trailing spaces is determined as defined in the following table.

Data type	Default operation
nchar	Pad original value with trailing blanks to the length of the column.
nvarchar	Trailing blanks in character values inserted into **nvarchar** columns are not trimmed. Values are not padded to the length of the column.

- If an INSERT statement violates a constraint or rule or if it has a value incompatible with the data type of the column, the statement fails and SQL Server CE displays an error message.
- If INSERT is loading multiple rows with SELECT, any violation of a rule or constraint that occurs from the values being loaded causes the entire statement to be terminated, and no rows are loaded.

Examples

A. Using a simple INSERT statement

The following example adds a new company to the **Customers** table in the **Northwind** database. Where certain information is unavailable, a null value is inserted.

```
INSERT INTO Customers VALUES ('TSTCU', 'Testing Site
Telephony Co.', 'John Kay', 'Owner', NULL, 'Forks',
NULL, NULL, 'USA', NULL, DEFAULT)
```

B. Inserting data that is not in the same order as the columns

The following example uses *column_list* and the VALUES list to specify explicitly the values that are inserted into each column in the **Customers** table in the **Northwind** database.

```
INSERT INTO Customers (CustomerID, CompanyName,
Country, Phone) VALUES ('XYZAB', 'Xylophone Alphabet
Co.', 'USA', '206-321-8765')
```

C. Inserting data with fewer values than columns

The following example adds Liz Smith to the **Employees** table in the **Northwind** database without supplying a value for **EmployeeID**.

```
INSERT Employees (LastName, FirstName) VALUES
('Smith', 'Liz')
```

Operators

Microsoft® SQL Server™ 2000 Windows® CE Edition (SQL Server CE) supports the following operators.

Arithmetic Operators

+(ADD)	/ (Divide)
- (Subtract)	% (Modulo)
* (Multiply)	

Bitwise Operators

Note: Only one expression can be of either binary or varbinary data type in a bitwise operation.

& (AND)	^ (Exclusive OR)
| (OR)	~ (NOT)

Comparison Operators

= (Equals)	<> (Not Equal To)
> (Greater Than)	!= (Not Equal To)
< (Less Than)	!< (Not Less Than)
>= (Greater Than or Equal To)	!> (Not Greater Than)
<= (Less Than or Equal To)	

Logical Operators

ALL	IN
AND	LIKE
ANY	NOT
BETWEEN	OR
EXISTS	SOME

Unary Operators

+ (Positive)	- (Negative)

ORDER BY Clause

Specifies the sort order for the result set. The ORDER BY clause is not valid in subqueries.

Refer to the example provided in **Select.zip**.

Syntax

```
[ ORDER BY { order_by_expression [ ASC | DESC ] }
[ ,...n] ]
```

Arguments

order_by_expression

Specifies a column on which to sort. A sort column can be specified as a name or column alias (which can be qualified by the table name) or an expression. Multiple sort columns can be specified. The sequence of the sort columns in the ORDER BY clause defines the organization of the sorted result set.

The ORDER BY clause can include items not appearing in the select list.

ASC

Specifies that the values in the specified column should be sorted in ascending order, from lowest value to highest value.

DESC

Specifies that the values in the specified column should be sorted in descending order, from highest value to lowest value. Null values are treated as the lowest possible values.

There is no limit to the number of items in the ORDER BY clause.

Examples

The following example lists employees by their first names.

```
SELECT FirstName + ' ' + LastName FROM Employees ORDER
BY FirstName
```

Reserved Words

A reserved word in Microsoft® SQL Server™ 2000 Windows® CE Edition (SQL Server CE) does not necessarily function the same way as the corresponding word in SQL Server 2000. The following table lists the reserved words used in SQL Server CE.

Important: Avoid using reserved words as identifiers. If reserved words must be used as identifiers, they must be delimited with double quotation marks.

@@IDENTITY	ENCRYPTION	ORDER
ADD	END	OUTER
ALL	ERRLVL	OVER
ALTER	ESCAPE	PERCENT
AND	EXCEPT	PLAN
ANY	EXEC	PRECISION
AS	EXECUTE	PRIMARY
ASC	EXISTS	PRINT
AUTHORIZATION	EXIT	PROC

AVG	EXPRESSION	PROCEDURE
BACKUP	FETCH	PUBLIC
BEGIN	FILE	RAISERROR
BETWEEN	FILLFACTOR	READ
BREAK	FOR	READTEXT
BROWSE	FOREIGN	RECONFIGURE
BULK	FREETEXT	REFERENCES
BY	FREETEXTTABLE	REPLICATION
CASCADE	FROM	RESTORE
CASE	FULL	RESTRICT
CHECK	FUNCTION	RETURN
CHECKPOINT	GOTO	REVOKE
CLOSE	GRANT	RIGHT
CLUSTERED	GROUP	ROLLBACK
COALESCE	HAVING	ROWCOUNT
COLLATE	HOLDLOCK	ROWGUIDCOL
COLUMN	IDENTITY	RULE
COMMIT	IDENTITY_INSERT	SAVE
COMPUTE	IDENTITYCOL	SCHEMA
CONSTRAINT	IF	SELECT
CONTAINS	IN	SESSION_USER
CONTAINSTABLE	INDEX	SET
CONTINUE	INNER	SETUSER
CONVERT	INSERT	SHUTDOWN
COUNT	INTERSECT	SOME
CREATE	INTO	STATISTICS
CROSS	IS	SUM
CURRENT	JOIN	SYSTEM_USER
CURRENT_DATE	KEY	TABLE
CURRENT_TIME	KILL	TEXTSIZE
CURRENT_TIMESTAMP	LEFT	THEN
CURRENT_USER	LIKE	TO
CURSOR	LINENO	TOP
DATABASE	LOAD	TRAN
DATABASEPASSWORD	MAX	TRANSACTION
DATEADD	MIN	TRIGGER
DATEDIFF	NATIONAL	TRUNCATE

DATENAME	NOCHECK	TSEQUAL
DATEPART	NONCLUSTERED	UNION
DBCC	NOT	UNIQUE
DEALLOCATE	NULL	UPDATE
DECLARE	NULLIF	UPDATETEXT
DEFAULT	OF	USE
DELETE	OFF	USER
DENY	OFFSETS	VALUES
DESC	ON	VARYING
DISK	OPEN	VIEW
DISTINCT	OPENDATASOURCE	WAITFOR
DISTRIBUTED	OPENQUERY	WHEN
DOUBLE	OPENROWSET	WHERE
DROP	OPENXML	WHILE
DUMP	OPTION	WITH
ELSE	OR	WRITETEXT

SELECT Statement

Retrieves rows from the database and allows the selection of one or many rows or columns from one or many tables. This is the primary SQL construct used to express queries. SELECT does not modify, insert, or delete any data.

Refer to the example provided in **Select.zip**.

Syntax

The main clauses of a SELECT statement are:

```
SELECT select_list
FROM table_source
[ WHERE search_condition ]
[ GROUP BY group_by_expression ]
[ HAVING search_condition ]
[ ORDER BY order_expression [ ASC | DESC ] ]
```

Remarks

A SELECT statement describes a query to the system. The execution of the query does not update any data. The query result is a table with identically structured rows, each of which has the same set of columns. The SELECT statement defines exactly which columns will exist in this result table, and the rows that will populate the table. The SELECT statement does not tell the system how to execute the query; instead, the system executes the query in whatever manner is deemed optimal (using an internal cost-based optimization module). The result is guaranteed to be equivalent to the following canonical execution strategy. The only differences may be in the order of rows in the table, although this will be consistent with any ordering specified by an ORDER BY clause.

Execution Strategy

1. Generate the join of tables in the FROM clause. If the explicit JOIN syntax is used, the JOIN result is obvious. If the FROM clause has a list of table names separated by commas, this is implicitly a cross-product join of the tables.
2. If a WHERE clause exists, apply the search condition to the rows resulting from Step 1, and retain only those rows that satisfy the condition.
3. If there are no aggregates in the SELECT clause, and if there is no GROUP BY clause, go to Step 7.
4. If there is a GROUP BY clause, divide the rows resulting from Step 2 into several groups, such that all the rows in each group have the same value on all the grouping columns. If there is no GROUP BY clause, put all the rows into a single group.
5. For each group arising from Step 4, apply the HAVING clause, if it is specified. Only those groups that satisfy the HAVING clause will be retained.
6. For each group arising from Step 5, generate exactly one result row by evaluating the select list from the SELECT clause against that group.
7. If the SELECT clause has the DISTINCT keyword, eliminate any duplicate rows in the result of Step 6.
8. If there is an ORDER BY clause, sort the result of Step 7 as specified by the order expression.

UPDATE

Modifies existing data in a table in Microsoft® SQL Server™ 2000 Windows® CE Edition (SQL Server CE).

Refer to the example provided in **Update.zip**.

Syntax

```
UPDATE table_name
   SET
   { column_name = { expression | DEFAULT | NULL } } [
,...n ]
   [ WHERE < search_condition > ]
```

Arguments

table_name

Is the name of the table to update.

SET

Specifies the list of column or variable names to be updated.

column_name

Is a column that contains the data to be changed. *column_name* must reside in the specified table and should be specified only once in the SET clause.

expression

Is a variable, literal value, or expression that returns a single value. The value returned by expression replaces the existing value in *column_name*.

DEFAULT

Specifies that the default value defined for the column is to replace the existing value in the column. This can also be used to change the column to NULL if the column has no default and is defined to allow null values.

WHERE

Specifies the conditions that limit the rows that are updated.

< search_condition >

Specifies the condition to be met for the rows to be updated. There is no limit to the number of predicates that can be included in a search condition.

Remarks

Identity columns cannot be updated.

If a WHERE clause is not specified, all rows of the table are updated.

The search condition in the WHERE clause is evaluated for each row of the table before updating any row of the table.

If an update to a row violates a constraint or rule, if it violates the NULL setting for the column, or if the new value is an incompatible data type, the statement is canceled, an error is returned, and no records are updated.

All **nchar** columns are right-padded to the defined length.

All trailing spaces are removed from data inserted into **nvarchar** columns, except in strings containing only spaces. These strings are truncated to an empty string.

Examples

The following example changes the shipping address in the **Orders** table of the **Northwind** database for all orders made by the company with the customer ID of VINET.

```
UPDATE Orders SET ShipAddress = '21 rue de
l''xylophie' WHERE CustomerID = 'VINET'
```

WHERE Clause

Specifies a search condition to restrict the rows returned.

Refer to the example provided in **Select.zip**.

Syntax

```
[ WHERE < search_condition >]
```

Arguments

< search_condition >

Restricts the rows returned in the result set through the use of predicates. There is no limit to the number of predicates, separated by an AND clause, that can be included in a search condition.

Examples

The following example uses the WHERE clause to get the total number of units in stock for all discontinued products in the Northwind database.

```
SELECT * FROM Products WHERE Discontinued = 'True'
```

Appendix C Maximum and Minimum Values for TaskExpert

The following table indicates the maximum and minimum values for TaskExpert items in the IND780 and IND560, respectively.

	IND780	IND560
Maximum Number of Lines	10000	5000
Maximum Program Size (bytes)	400008	102408
Default Program Size (bytes)	1024	1024
Maximum String Length (characters)	1001*	1001
Maximum Length for OPEN COM (characters)	250	250
*Maximum string length is 500 in IND780 version 6.1.01 firmware due to UNICODE support		
Maximum PLC Length (bytes)	500	100
Maximum Length for TaskGlobal or Global Variable Name	16	16
Maximum Dimensions for Array	3	3
Maximum Number of Variables	2048	1000
Maximum Program Name Length	50	8
Maximum Filename Length	80	80
Maximum Appended Name Length	21	13
Maximum Database Name Length	10	N/A
Maximum Number of Concurrently Open Databases	4	N/A
Number of Standard Tables	10	10
Maximum Number of Softkeys	15	15
Maximum Number of Application Keys	4	N/A
Maximum Number of SD Instances	15	15
Maximum Number of GOSUB Levels	20	20
Maximum Number of WHILE Levels	15	20
Maximum Number of FOR Levels	15	20
Maximum Number of Elements in Expression Stack	64	64
Maximum Number of Debug Break Lines	50	50
Maximum Number of TCP/IP Sockets	12	12
Maximum Number of Input Keys	100	100
Maximum Number of TE Display Objects	50	20
Maximum Length of Display Message	160	160

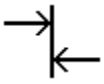
Appendix D IND560 and IND780 Softkeys

IND560 Softkeys

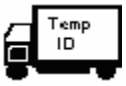
Function	Softkey ID	Graphic Filename	Graphic Image
Adjust Contrast	1	contrast.bmp	
Alibi	2	alibi.bmp	Alibi
Calibration Test	3	cal_test.bmp	
Recall Info	4	recall.bmp	
Reports	5	reports.bmp	
Setup	6	setup.bmp	
SmartTrac	7	Sm_trac.bmp	Smart -Trac
Tare Table	8	tare_mem.bmp	
Target	9	target.bmp	
Target Control	10	control.bmp	
Target Table	11	targ_mem.bmp	
Target Start	12	start.bmp	
Time & Date	13	timedate.bmp	
Unit Switching	14	select.bmp	
X10 Display	15	x10.bmp	x10
ID	19	id.bmp	ID

Function	Softkey ID	Graphic Filename	Graphic Image
MinWeigh	31	minweigh.bmp	
Container Tare	16	cntnr_tr.bmp	
Cycles	17	cycles.bmp	
Formula	18	formula.bmp	
Weigh-in Start	20	weigh_in.bmp	
Weigh-out Start	21	weighout.bmp	
Container Tare Table	35	cntnr_m.bmp	
Target Weigh-in	33	targ_in.bmp	
Target Weigh-out	34	targ_out.bmp	
Comparators	39	comprtr.bmp	
Custom Trigger 1	40	trigger1.bmp	
Custom Trigger 2	41	trigger2.bmp	
Custom Trigger 3	42	trigger3.bmp	

IND780 Softkeys

Soft Key Name	Softkey ID	Graphic Filename	Graphic Image
Adjust Contrast	8356	contrast.bmp	
Alibi Memory	8171	alibi.bmp	
Calibration Test	8196	cal_test.bmp	
Comparators	8657	comparator.bmp	

Soft Key Name	Softkey ID	Graphic Filename	Graphic Image
Custom Trigger 1	24	CustomTrigger1.bmp	
Custom Trigger 2	25	CustomTrigger2.bmp	
Custom Trigger 3	26	CustomTrigger3.bmp	
Custom Trigger 4	27	CustomTrigger4.bmp	
Custom Trigger 5	28	CustomTrigger5.bmp	
ID1	8661	ID1.bmp	
ID2	8662	ID2.bmp	
MinWeigh	20	minweigh.bmp	
Recall Info	8362	info.bmp	
Repeat Print	29	repeat_print.bmp	
Reports	8372	reports.bmp	
Select Terminal	9	terminal4.bmp	
Setup	1	setparam.bmp	
SmartTrac	10	SmartTrac.bmp	
Start Task 1	21	StartTask1.bmp	
Start Task 2	22	StartTask2.bmp	
Start Task 3	23	StartTask3.bmp	
Tare Table	8153	tare_table.bmp	

Soft Key Name	Softkey ID	Graphic Filename	Graphic Image
Target	12	setpoint.bmp	
Target Control	19	target_control.bmp	
Target Start	16	start.bmp	
Target Table	8150	target_table.bmp	
Task List	8403	clipboard.bmp	
Time & Date	8090	TD.bmp	
Unit Switching	14	select.bmp	
x10 Display	15	x10.bmp	
Index Table	203	index.bmp	
Manual Lights	208	manual lights.bmp	
Temporary ID	202	tempID.bmp	
Transient Vehicle	204	transveh.bmp	
Vehicle ID	201	vehID.bmp	
Container Table	170	Container_Tare_Memory.bmp	
Container Tare	160	Contain_Tare.bmp	
Formula	130	recipe.bmp	
Index Table	180	index.bmp	
Number of Cycles	100	cycles.bmp	

Soft Key Name	Softkey ID	Graphic Filename	Graphic Image
Start Weigh In	110	fill.bmp	
Start Weigh Out	120	discharge.bmp	
Target Weigh In	140	Target_In.bmp	
Target Weigh Out	150	Target_Out.bmp	

Appendix E Loading TaskExpert Files

IND780

The following procedure outlines installing the TaskExpert file(s) and bitmap images to the IND780.

The TaskExpert and bitmap (.bmp) files can only be loaded via FTP.

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 IND780 IP address and user (default: user **admin**, password **admin**).
3. Select the files to be uploaded to the terminal.
4. Press Upload.

Loading via FTP

1. Connect to IND780 using FTP utility (default: user **admin**, password **admin**).
2. Place the .cpt files into the directory /TaskExpert/Programs/.
3. Place the .bmp files into the /Terminal/SKBMP/(color or mono).

Once all the files are loaded, the application can be set either to Manual Start or Auto Start in the IND780 setup menu at **Application > TaskExpert > Start**.

IND560

Notes

- To run a TaskExpert application, the IND560 **must** have version 3.01 or higher firmware installed.
- In addition to the software, the IND560 **must** have a main board version (V0.8) installed. The version can be checked by viewing the System Info Recall screen on the IND560. Scroll down to the Hardware section and view the scale type. If there is a (V0.8) after the Analog L/C or IDNet text, the main board will support running a TaskExpert application. If the version is (V0.2) or there is no version shown, the main board must be replaced with a new version before running a TaskExpert application.

The following procedures outline methods used to install the TaskExpert file(s) and bitmap images to the IND560.

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

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 IND560 IP address and user (default: user admin password admin).
3. Select the files to be uploaded to the terminal.
4. Press Upload.

Loading via FTP

1. Connect to IND560 using FTP utility (default: user admin password admin).
2. Place files into the root directory (default FTP location).

Loading via Serial

1. Place IND560 into Test Mode (SW2-1 = ON).
2. Connect to IND560 using HyperTerminal. Settings for HyperTerminal should be:

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

3. Login to the IND560 shared data server (type: **user admin**).

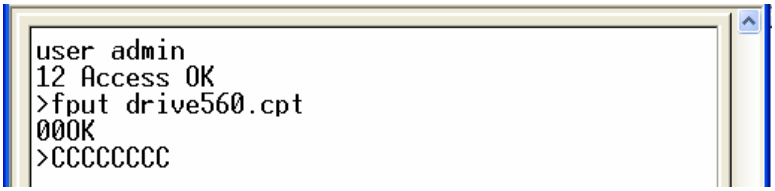


Figure E-1: IND560 Shared Data Server Login

4. Use the fput command to notify IND560 of the incoming file.

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

5. Once the "CCCC…" begins to display (Figure E-1), navigate to the Transfer > Send File menu in HyperTerminal.
6. Select the file to send – for example, **Drive560.cpt**.
7. Select the 1K Xmodem protocol.
8. Press Send. Once the file has been completely sent, the IND560 will return an <OK> back to HyperTerminal.
9. 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 IND560 setup menu at **Application > TaskExpert > Start**.

Appendix F Using IND560 Fonts

Ten configurations are available for setting fonts for display objects in the IND560 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 IND560 display when that font is used.

Tool Font	Height, in lines	# of Characters Possible
Alpha	1	21
Chinese	2	21
Num1	2	14
Num2	2	12
Num3	2	10
Num4	2	9
Num5	4	8 + DP
Num6	4	7 + DP
Num7	4	6 + DP
Num8	4	5 + DP

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 IND560 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.

Examples

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

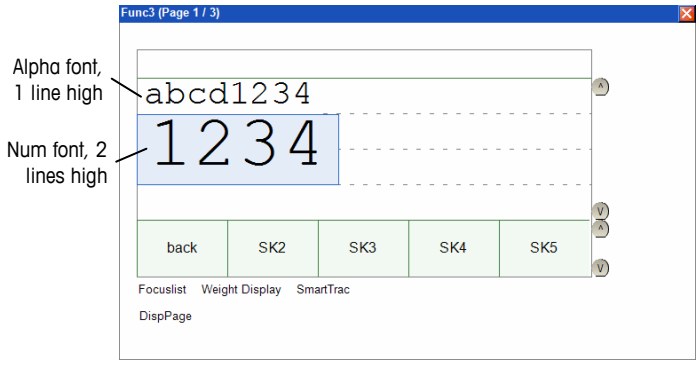


Figure F-1: Small (Alpha) and Medium (Num3) Fonts

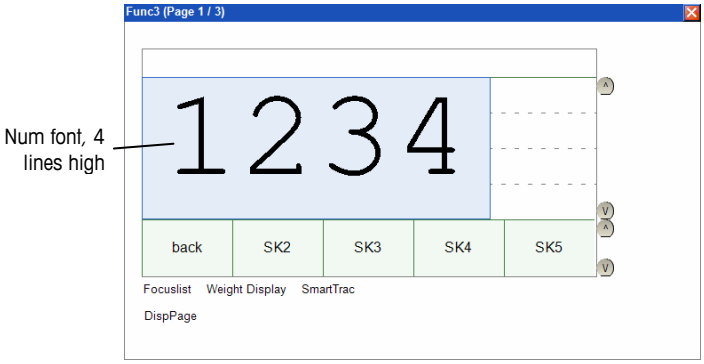


Figure F-2: Large (Num8) Font

Appendix G Capturing Screen Shots in the IND780

The TaskExpert PC tool contains a reporting feature that allows screen captures to be taken from the IND780.

Note: This viewer is 'read only' – it cannot be used to control the IND780 remotely. Any key presses required to navigate through the screens must be entered directly from the IND780 front panel.

Using Screen Capture

Opening the Viewer

Open the viewer from the Report menu (Figure 1).

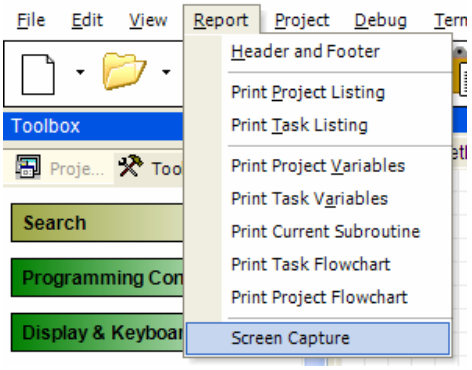


Figure 1: Report Menu Showing Screen Capture Option

Once connected, the viewer appears as a floating window on top of the PC Tool development environment (Figure 2).

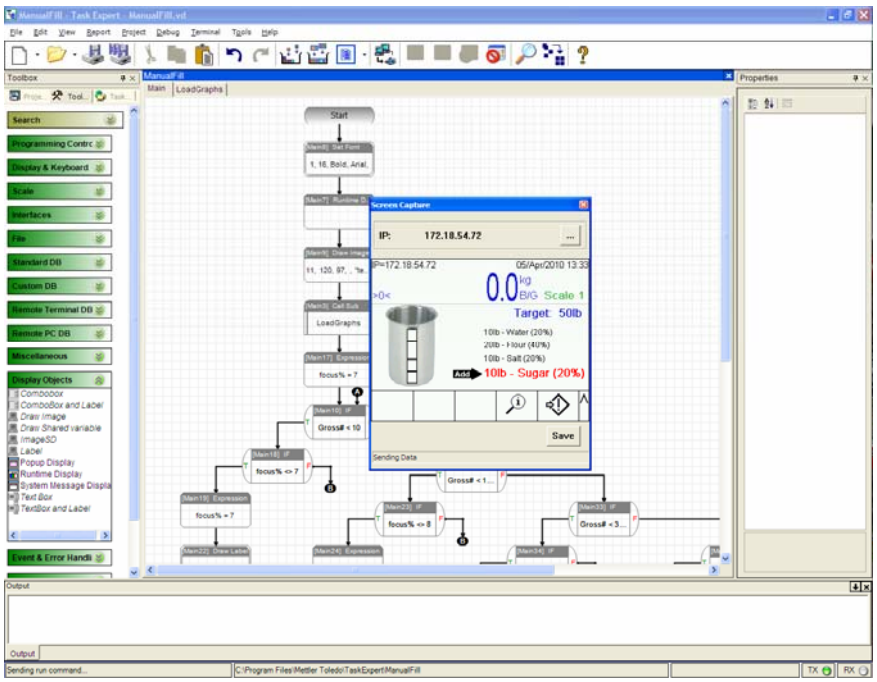


Figure 2: Screen Capture Viewer Displayed

Capturing and Saving Screen Images

The Screen Capture window (Figure 3) has two control buttons. Use the ellipsis (...) next to the **IP:** display at top to modify the IP address.

Note: When this button is pressed, the viewer automatically disconnects from the IND780.

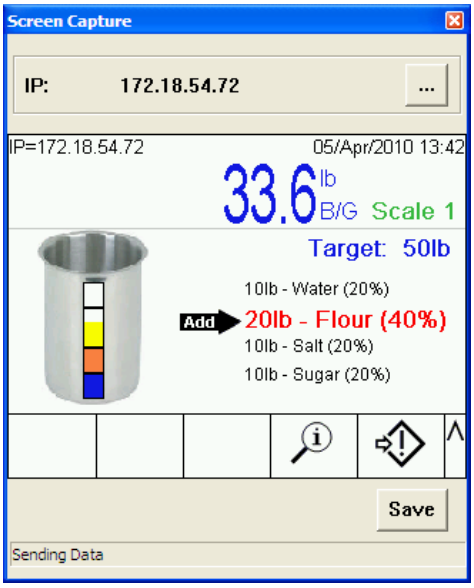


Figure 3: Screen Capture Window

The Save button opens a 'Save As' dialog window (Figure 4), in which the name and location of the screen capture image can be selected.

Images are automatically cropped to size and saved in a .jpg format for easy import into documentation.

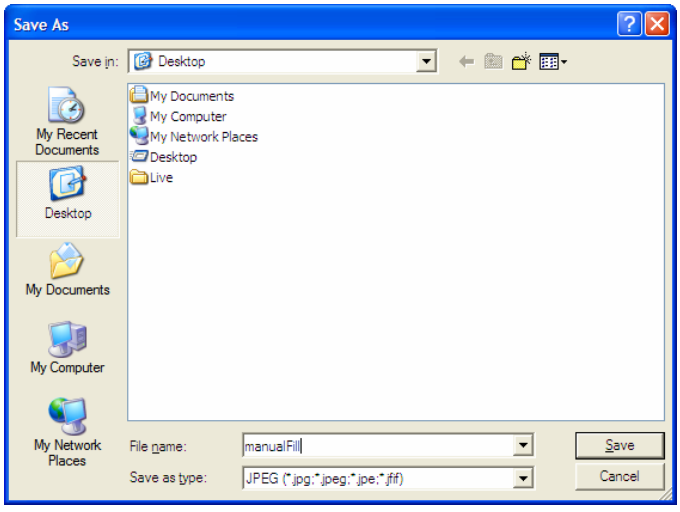


Figure 4: Screen Capture Save As... Dialog

Closing the Viewer

When the Screen Capture window is closed by clicking on the red **x**, the viewer disconnects from the IND780.

Appendix H Release Notes and Revision History

Document Revision	Firmware Version	Date	Changes
05	IND560: 4.02 IND560x: 3.02 IND560 PDX: 4.02 IND780: 6.5.15 (PXA255 board) IND780: 7.0.05 (PXA270 board)	06/2011	Added TaskExpert Libraries (Appendix A); added various graphing and array commands for IND780
04	IND560: 3.02 IND780 6.1.01 and higher	04/2010	Added Analog Output functions
03		01/2009	Added IND560/IND780 differentiation; modified font and color parameters; added IND560 Display Objects section; corrected errors
02	IND780: 5.1.04 or higher	12/2008	Restructured table and database chapters
01	–	05/2007	Updates to IND780 TaskExpert specification
00	–	11/2006	(First release)

METTLER TOLEDO
1900 Polaris Parkway
Columbus, Ohio 43240

64060431