QuickBuilder

An Introduction

Control Technology Corporation

ctc Control Technology Corp.

What Is QuickBuilder?

- QuickBuilder is a graphical, interactive development environment for CTC's Model 5300 series.
- QuickBuilder uses a block-diagram approach using Sequential Function Chart (SFC) to model and diagram logical tasks and program flow. It serves as an organizational tool for your entire project.

ctc
Control Technology Corp.

QuickBuilder Automation Suite

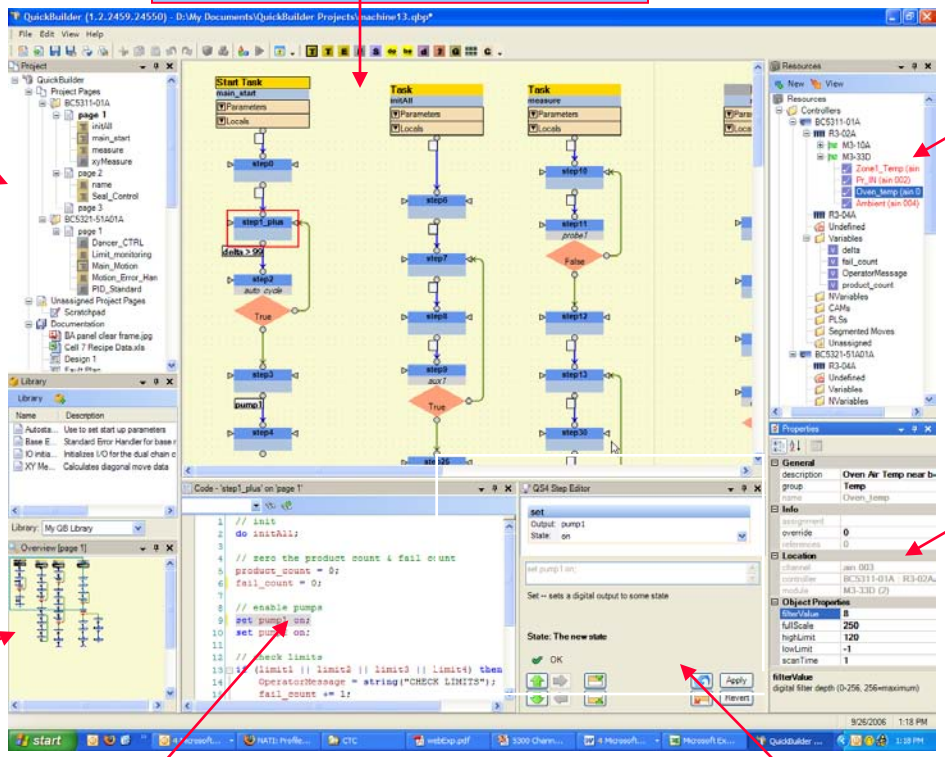
Graphical SFC Project / Editor Window

Project Manager / Tree List

Resource Manager

Library Manager

Property Inspector



Project Navigator

Step Code Editor

Automatic Step Builder

ctc Control Technology Corp.

QuickBuilder Automation Suite

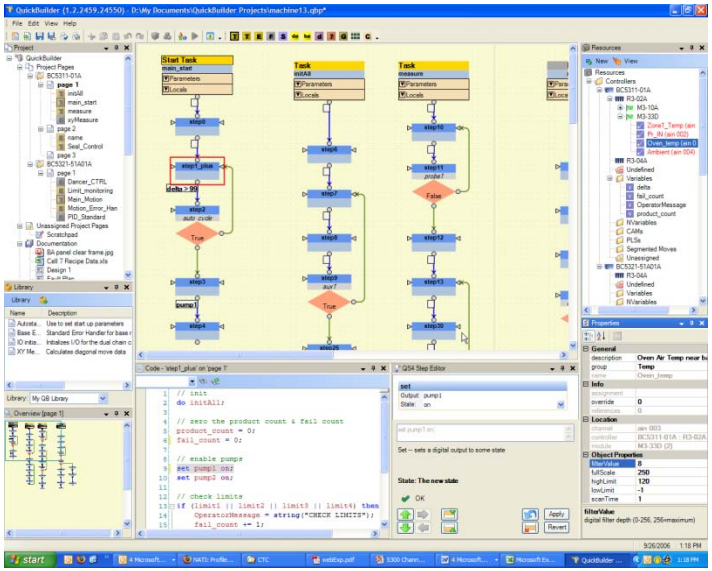
- The 5300 Graphical development environment

Configure

Program

Commission

Maintain



ctc Control Technology Corp.

Typical Use Scenario

User launches QuickBuilder

CONFIGURATION

- User adds one or more controllers
- User adds one or more racks
- User adds one or more I/O modules based on the anticipated or required configuration

PROGRAMMING

- User begins to define overall logic in SFC graphical form
- User codes required steps in the project or editor window

ctc Control Technology Corp.

Typical Use Scenario

COMMISSIONING

- User translates, corrects errors as required, and publishes the project for execution on the controller(s)
- User accesses the status/control window and one or more watch windows to debug the project

ctc Control Technology Corp.

Typical Use Scenario

MAINTAINING

- User auto-generates project documentation, including cross reference tables.
- User adds additional documentation using the internal word processor, or by attaching external files, such as spreadsheets, CAD files, diagrams, or photos.
- User remotely monitors the controller and updates or changes the active project over a secure network connection.

- *Note: All controller firmware can also be updated over the network*

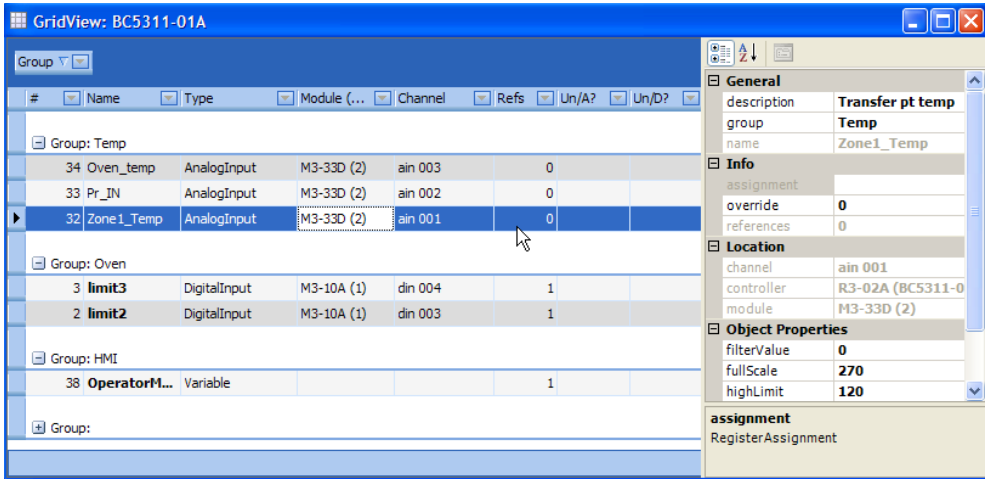
Control Technology Corp.

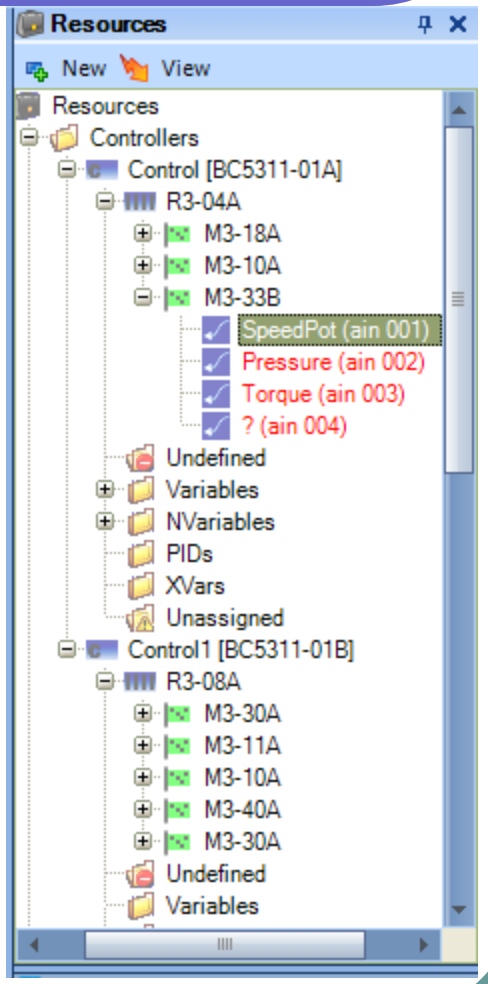
QuickBuilder

CONFIGURATION

Resource Manager

- Hierarchical view of all project resources
- Drag-and-drop functionality
- Right-click menus
- Tree view by type or connection
- Grid view for maximum flexibility



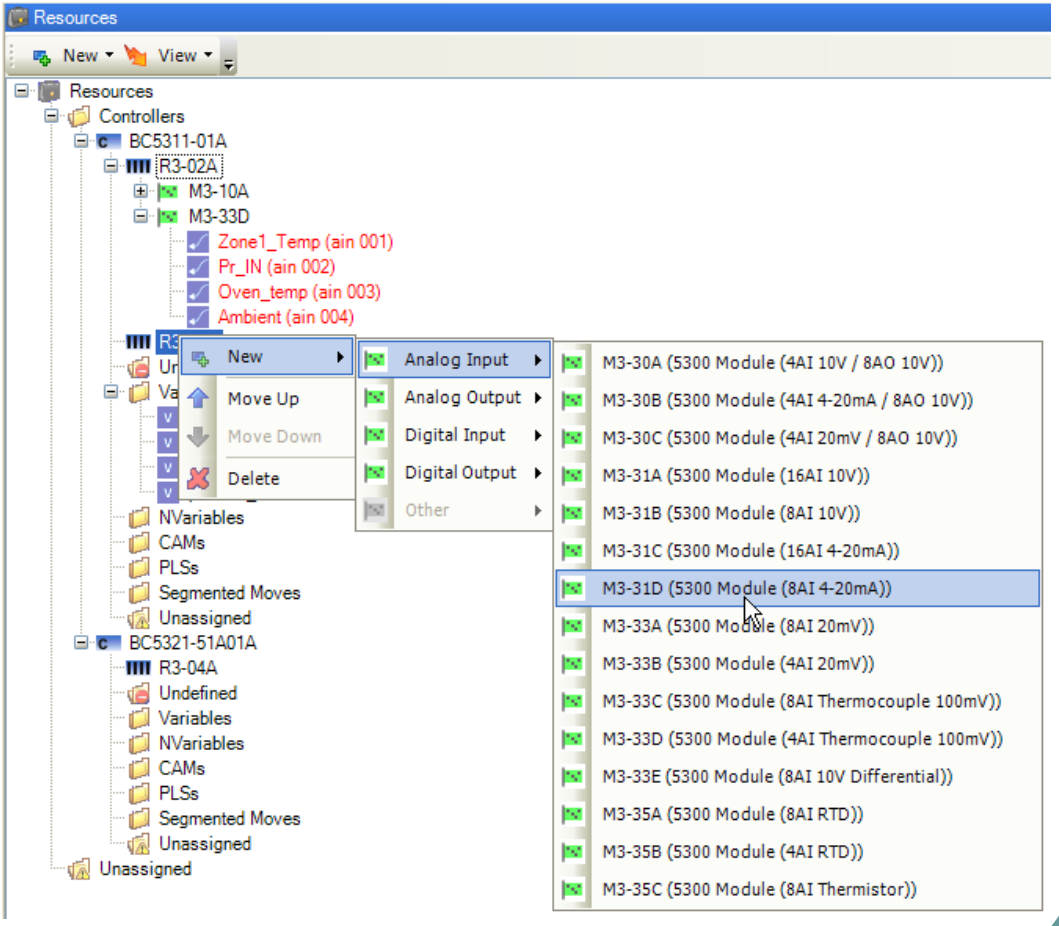


ctc Control Technology Corp.

QuickBuilder

CONFIGURATION

Resource Manager



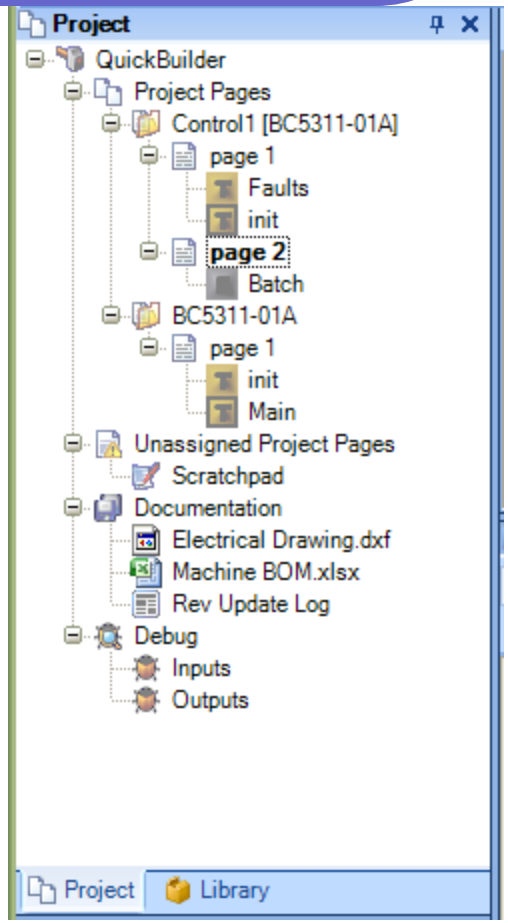
ctc Control Technology Corp.

QuickBuilder

CONFIGURATION

Project Manager

- Hierarchical view of major program elements
- Supports multi-controller projects
- Quickly drill down to specific tasks or functions
- Organizes project documentation
- Organizes debug views



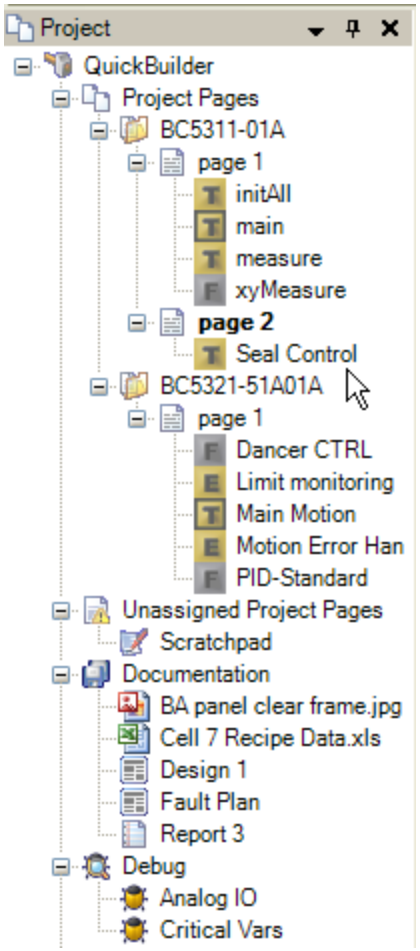
ctc Control Technology Corp.

QuickBuilder

CONFIGURATION

Project Manager

- Holds the logic, documentation and debug views for the project
  - Logic consists of a series of *pages* that hold diagrams of connected graphical constructs
  - Documentation includes external files (*links)* as well as documents authored through an integrated document editor
  - Debug Views
    - Watch and Modify variables
    - Load and Save *arrays* from/to CSV and the clipboard



ctc Control Technology Corp.

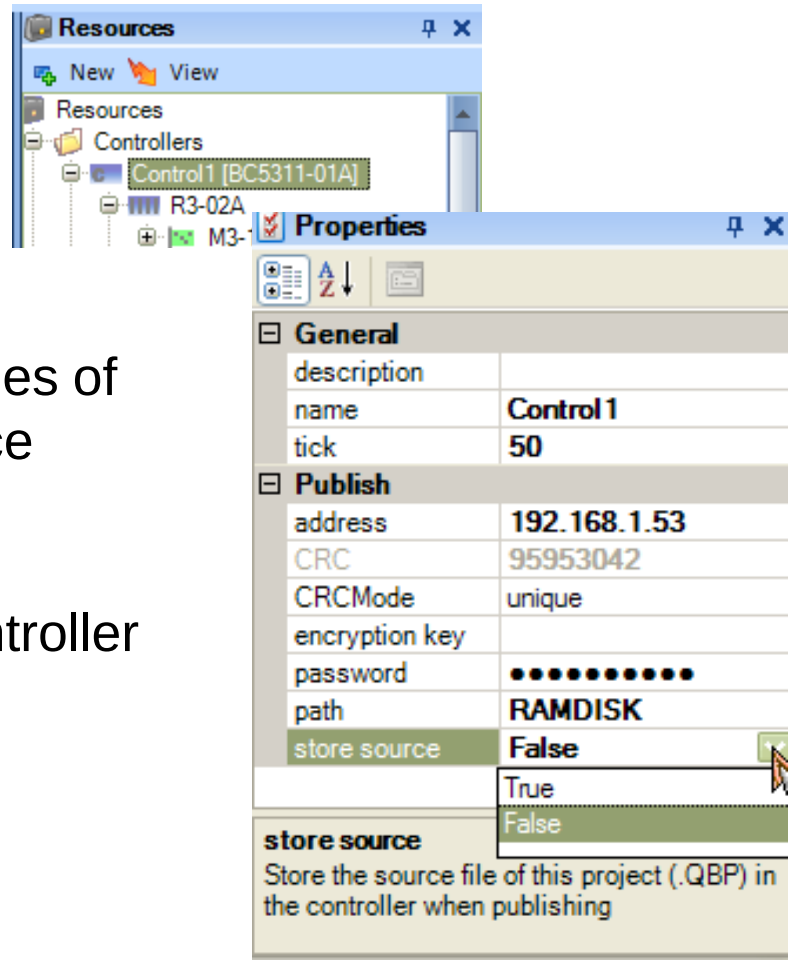
QuickBuilder

CONFIGURATION

Property Inspector

- View and edit detailed properties of the selected item in the resource manager.

- Set Initialization values for controller

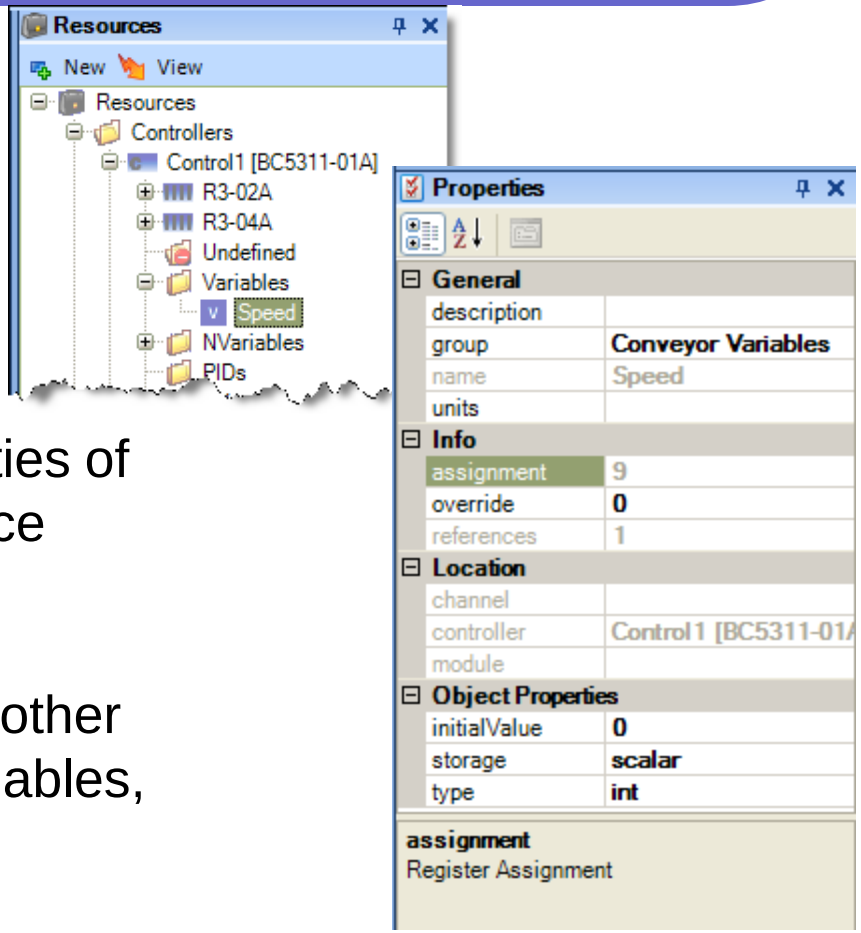


ctc Control Technology Corp.

QuickBuilder

CONFIGURATION

Property Inspector (cont.)



- View and edit detailed properties of the selected item in the resource manager.

- Set Initialization values for all other resources (inputs, outputs, variables, servos, etc.)

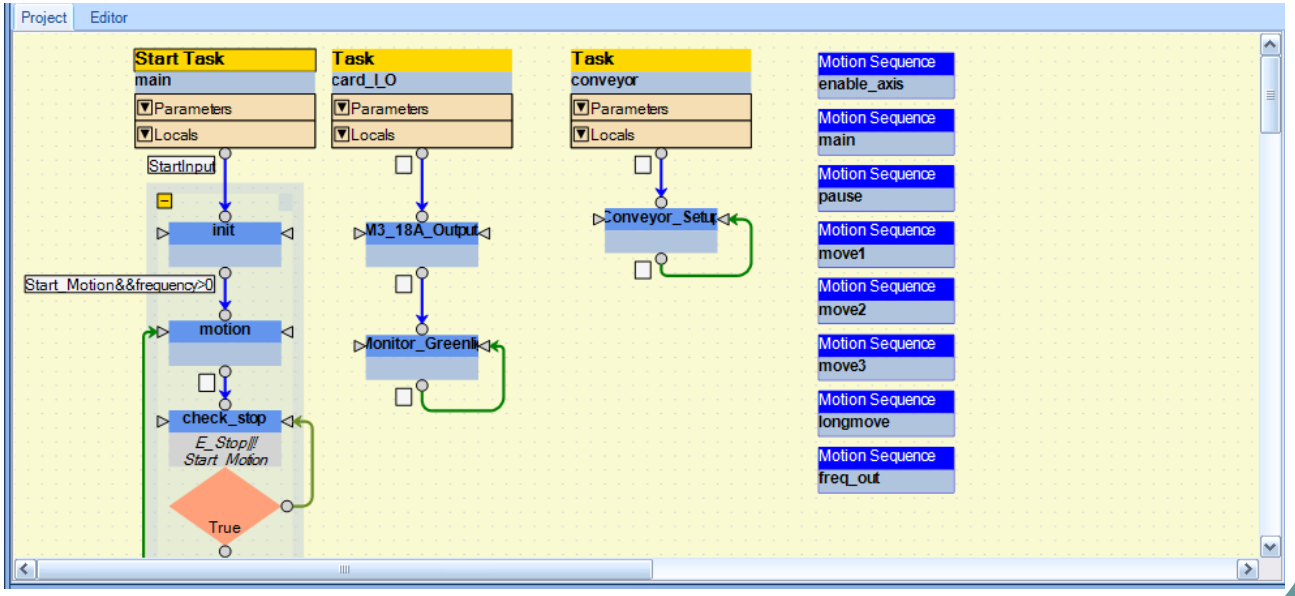
ctc Control Technology Corp.

QuickBuilder

PROGRAMMING

- The Project Window gives you a Graphical representation of the sequential flow control of the program.

- Divides the program into logical elements that are easy to manage and visualize.

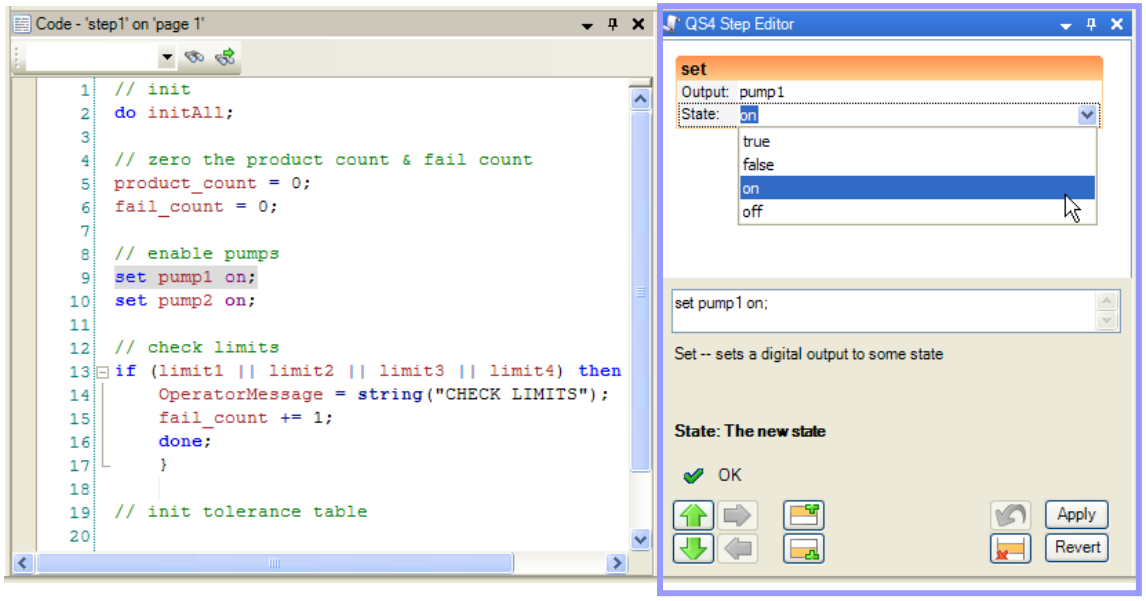


ctc Control Technology Corp.

QuickBuilder

PROGRAMMING

QuickStep 4 (QS4) is the primary programming language used in QuickBuilder applications. QS4 is the latest generation of CTC's state-based programming language. Now updated with floating point, the ability to handle advanced algorithms, strings, floats, complex branching, etc. Patterned after 1131 Structured text, but much easier to use!



The Step Editor allows beginners to quickly complete applications.

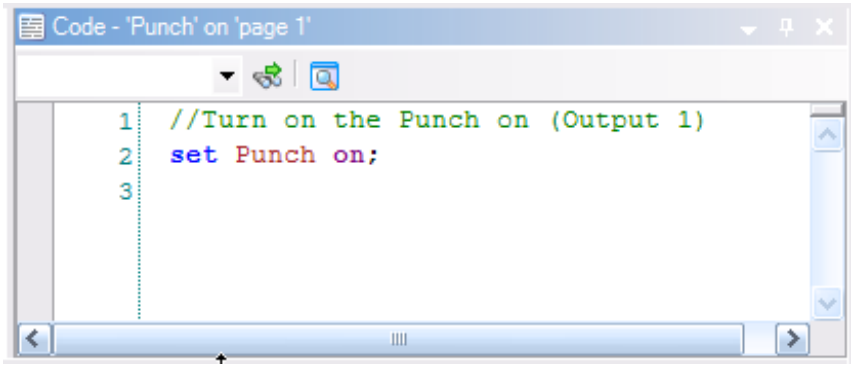
ctc Control Technology Corp.

QuickBuilder

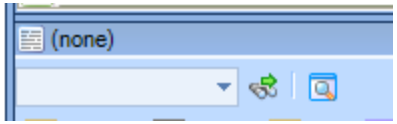
PROGRAMMING

Code / Comment Editor

- Used to enter and edit QuickStep 4 code and comments
- Syntax colored for readability
- Real-time error checking



- Quick-Find & Find



  - Quick-Find – Code/Comment editor search
  - Find – Project wide search

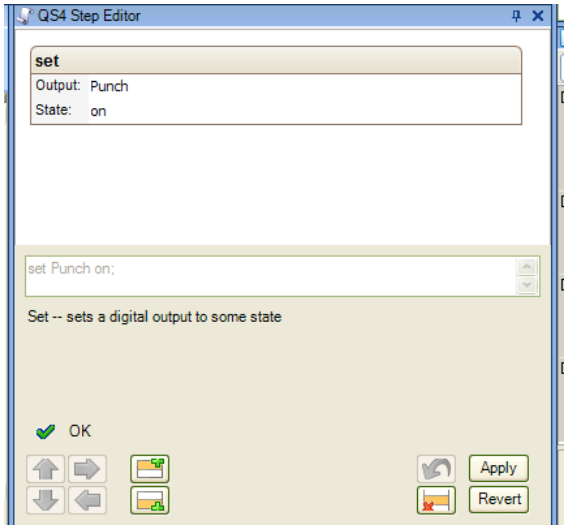
ctc Control Technology Corp.

QuickBuilder

PROGRAMMING

QS4 Step Editor

- Used to enter and edit code using the familiar “fill-in-the-blanks” method
  - Combo-boxes automatically fill in with relevant choices (such as the names of defined digital outputs when one is required)
- Allows new users to gain familiarity with the QS4 language and syntax
  - Easier transition for QS2 users
- Real-time error checking



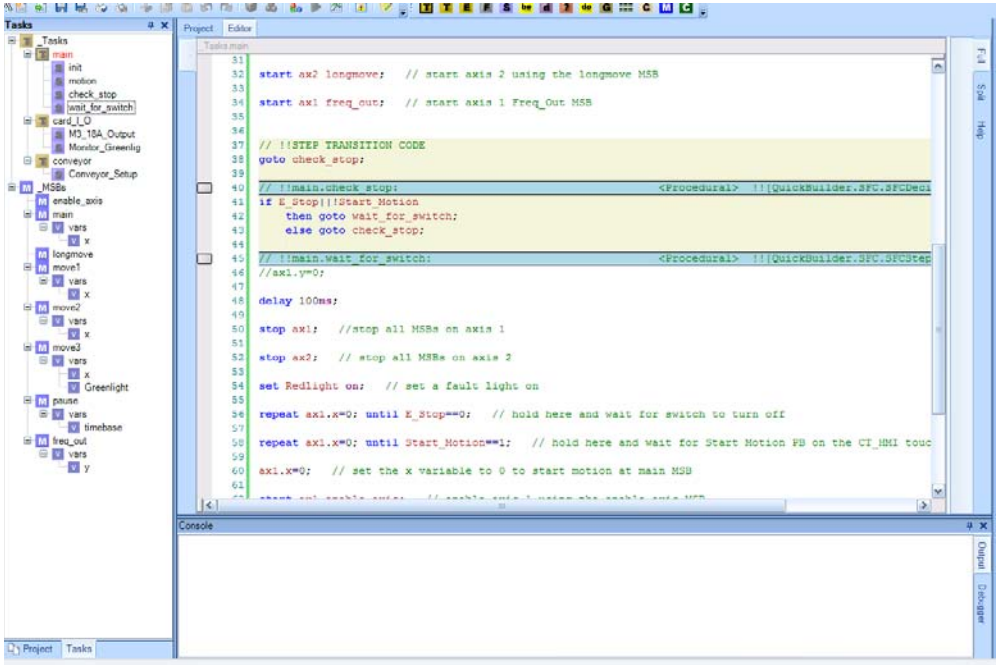
ctc Control Technology Corp.

QuickBuilder

PROGRAMMING

The Editor Window gives you a complete view of all the code within any selected task or MSB.

- Split-screen option allows you to look at two tasks or MSBs side by side.
- On demand help screen option gives you help as you code your project.



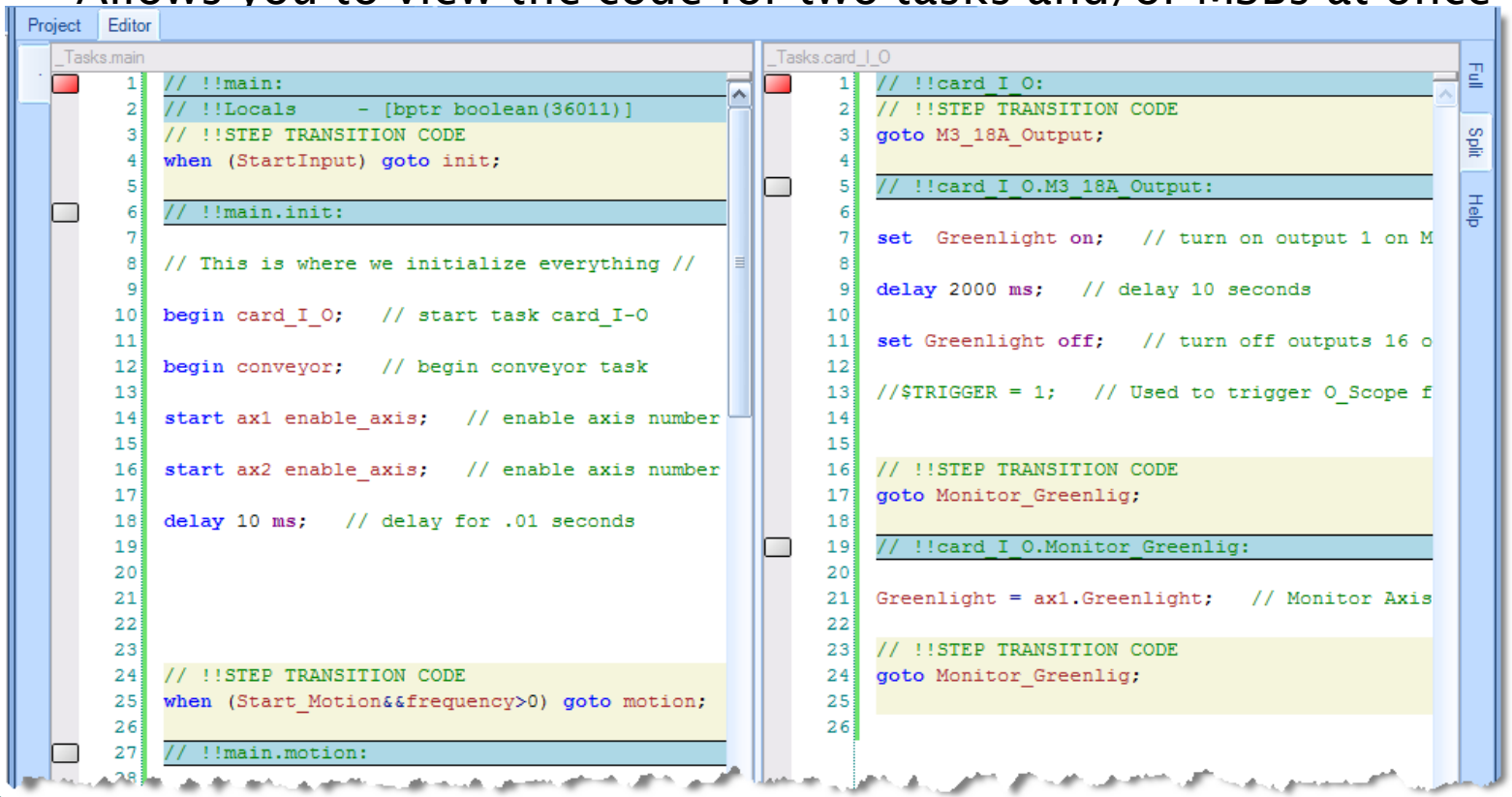
ctc Control Technology Corp.

QuickBuilder

PROGRAMMING

Editor Split Screen

- Allows you to view the code for two tasks and/or MSBs at once



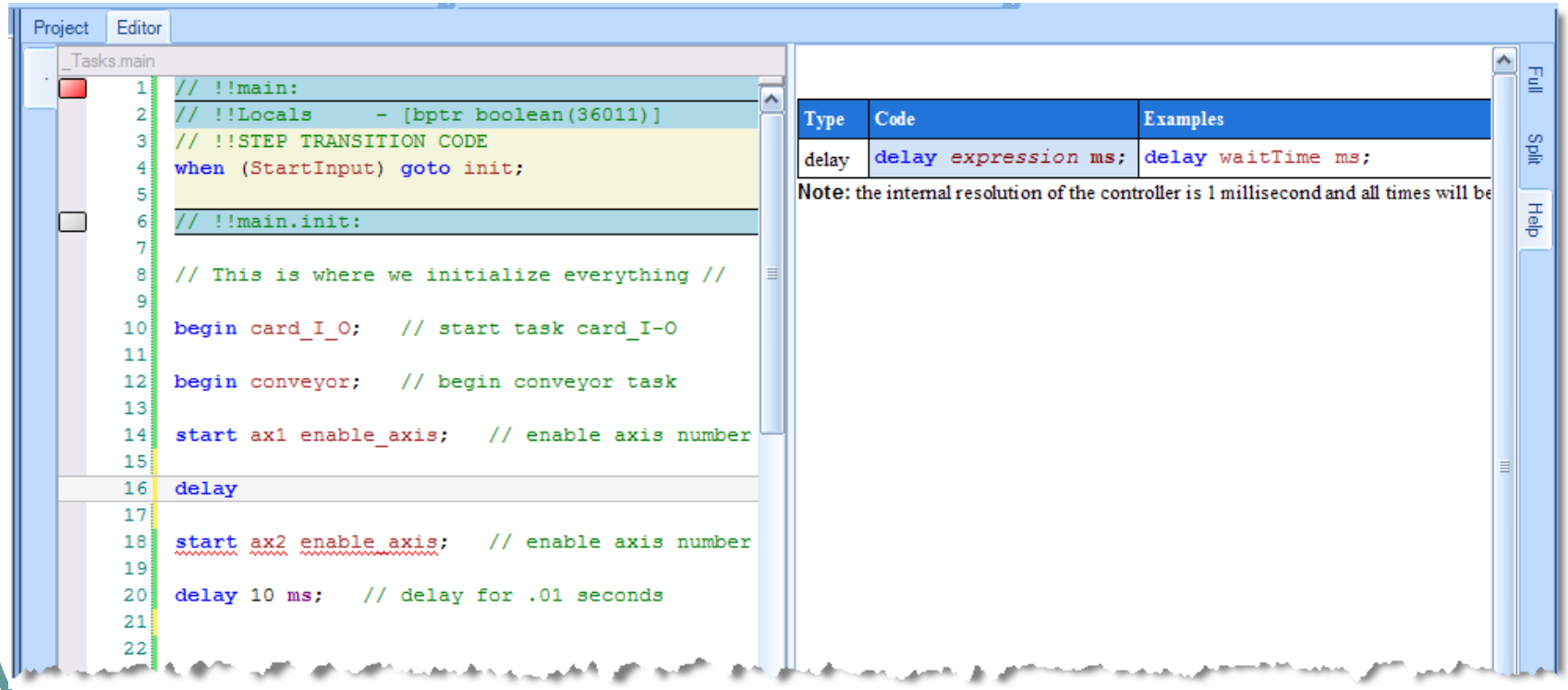
Control Technology Corp.

QuickBuilder

PROGRAMMING

Editor Help Screen

- Brings up code information with examples as you code your program



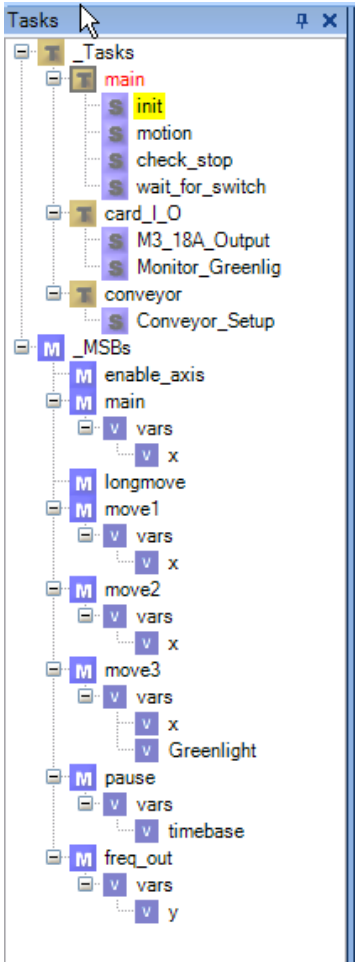
ctc Control Technology Corp.

QuickBuilder

PROGRAMMING

Tasks Window

- Automatically available while the Editor Window is open
- Let's you quickly select which task(s) or MSB(s) are open in the Editor Window
- The currently selected Task or MSB is highlighted



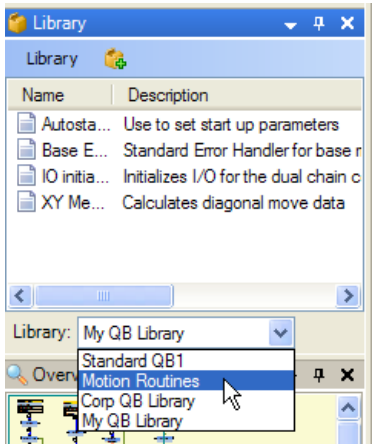
Control Technology Corp.

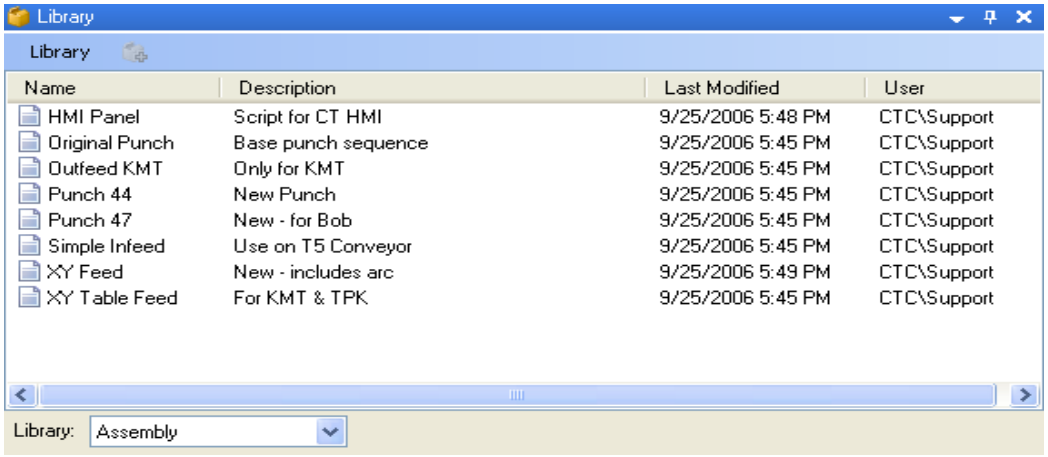
QuickBuilder

PROGRAMMING

Librarian

QuickBuilder's powerful library manager allows individuals and teams to easily share and re-use common code elements. Any portion of a project — a step, a series of steps, a task or an entire page — can be saved as a library element. Library elements are stored in a folder that can be located on the local PC or shared server.





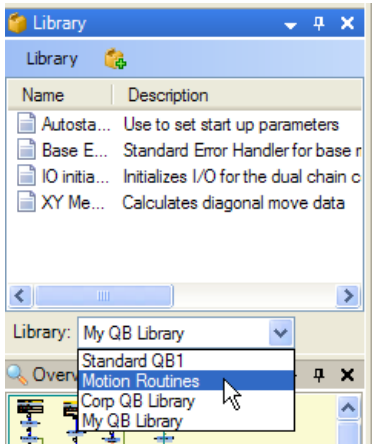
ctc Control Technology Corp.

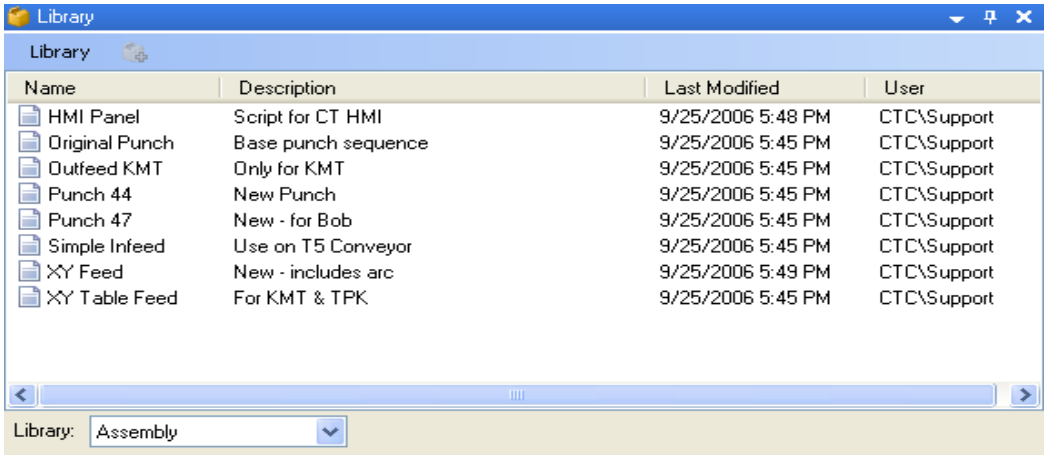
QuickBuilder

PROGRAMMING

Librarian

- Stores and retrieves snippets of logic
- Multiple libraries are supported.
- Encourages code re-use
- Minimizes debug time when proven logic is inserted into a new project
- Standardize Projects and Programs
- Corporate (networked) Libraries as well as User Libraries are supported





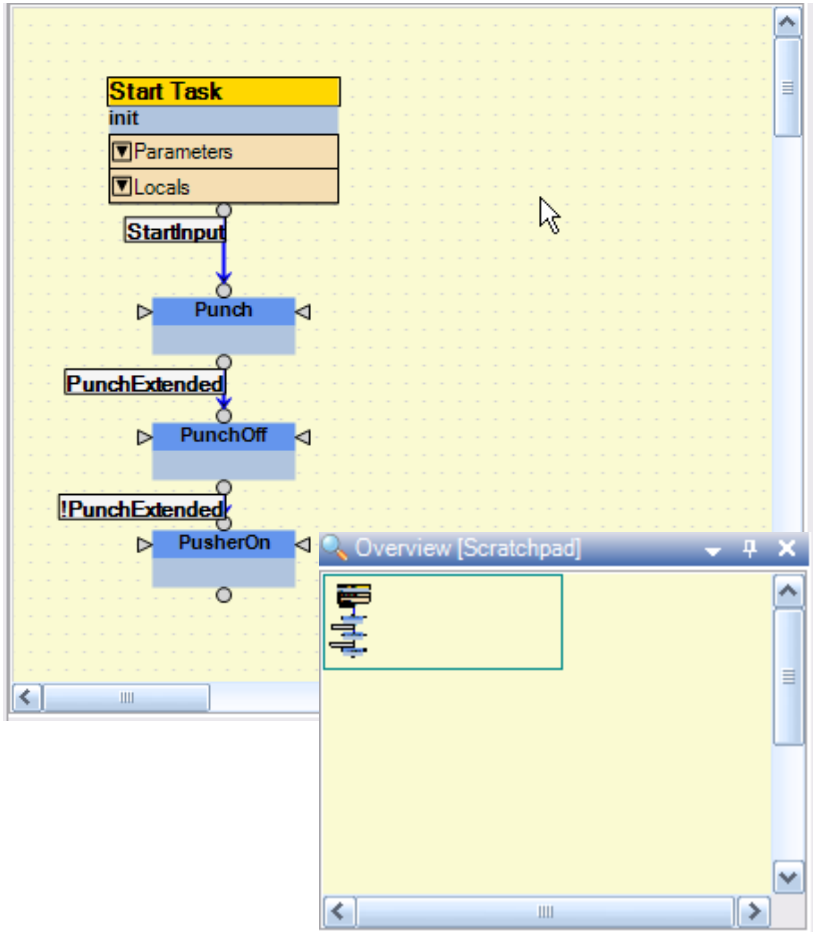
ctc Control Technology Corp.

QuickBuilder

PROGRAMMING

SFC and Overview Views

- Used to create and edit the overall logic for the project
- Familiar flowchart-style graphical programming



ctc Control Technology Corp.

QuickBuilder

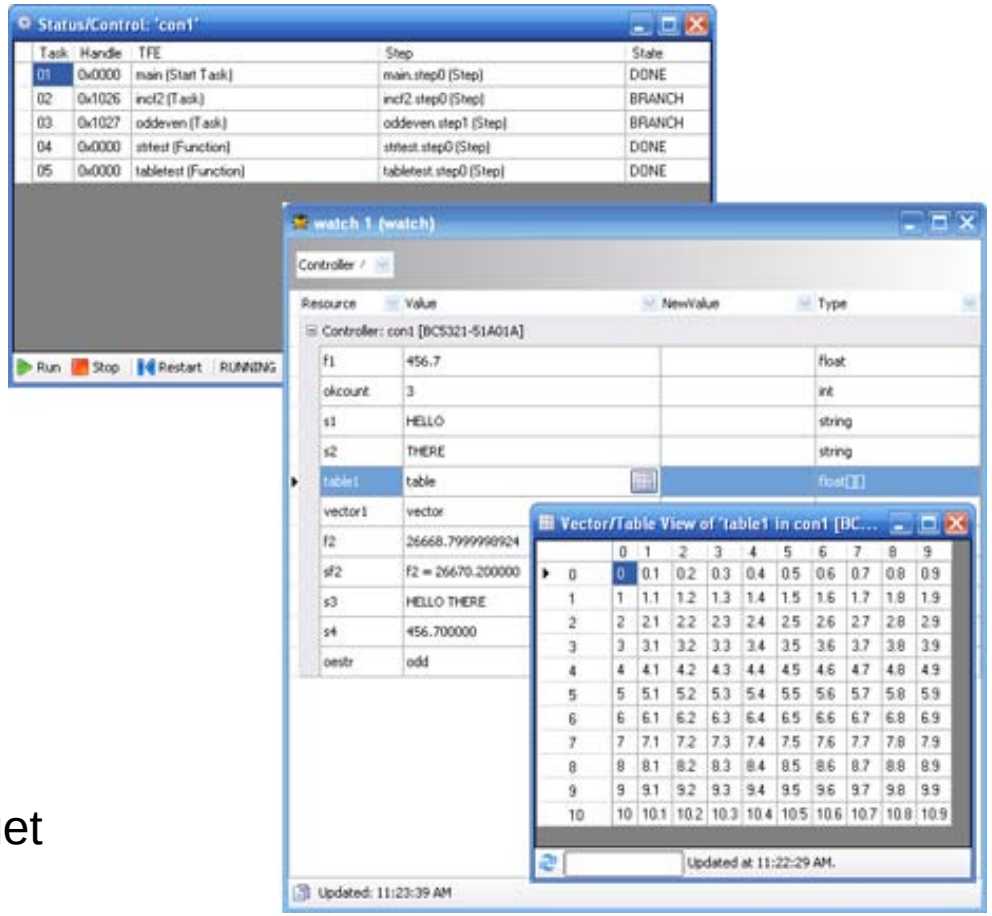
COMMISSIONING

Test & Debug tools

Status/Control Window
- Program flow control

Watch Windows
- Multiple watch windows
- System state capture to file
- Array builder
- Change resource values
- Data Logging

Editor
- Online option allows you to get “oneshot” values of resources by simply mousing over them.



ctc Control Technology Corp.

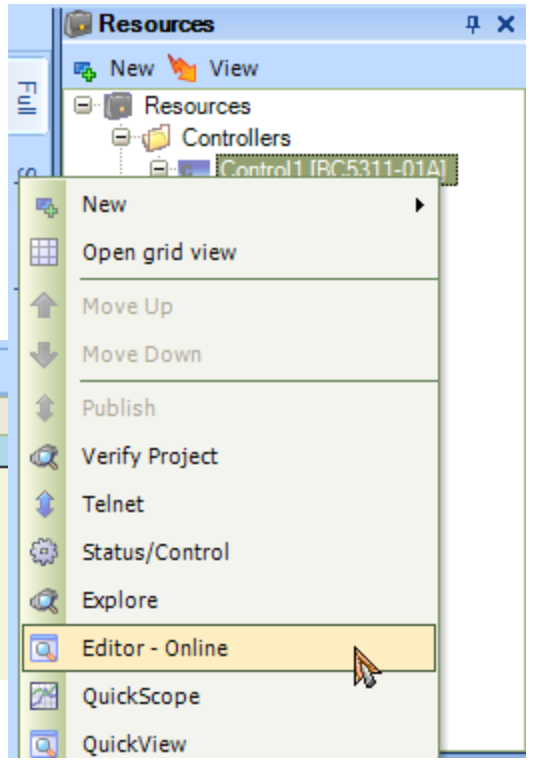
QuickBuilder

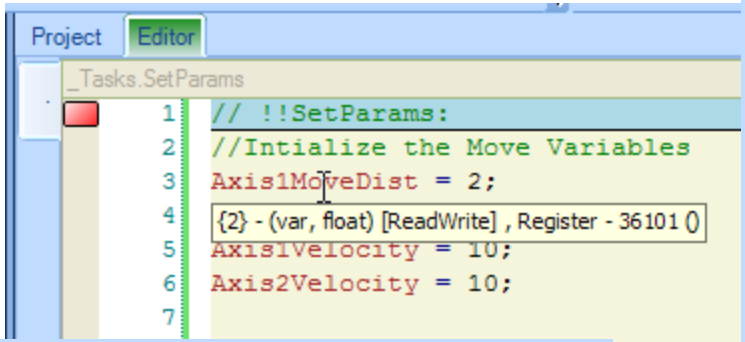
COMMISSIONING

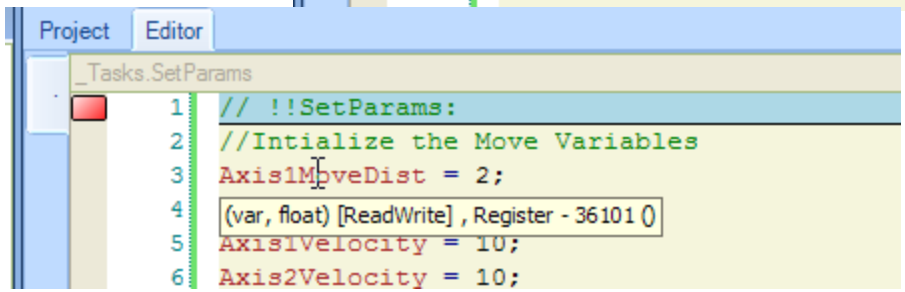
Test & Debug tools

Editor

- Online option allows you to get “oneshot” values of resources by simply mousing over them.





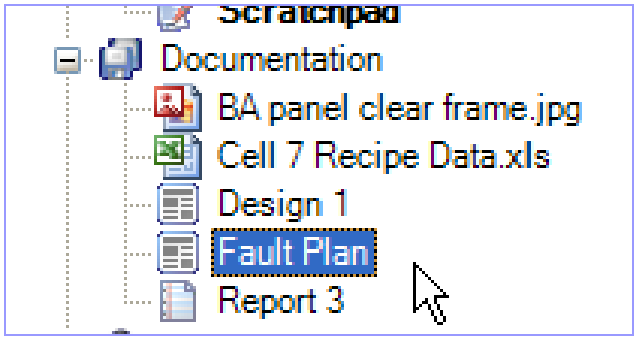


- You can also see offline info at any time

ctc Control Technology Corp.

QuickBuilder

MAINTAINING



- Detailed auto report generation for system documentation, revision control, password protection

- Ability to upgrade system and module firmware via the network connection

- Monitoring tools, remote diagnostics, logging, email exception reporting

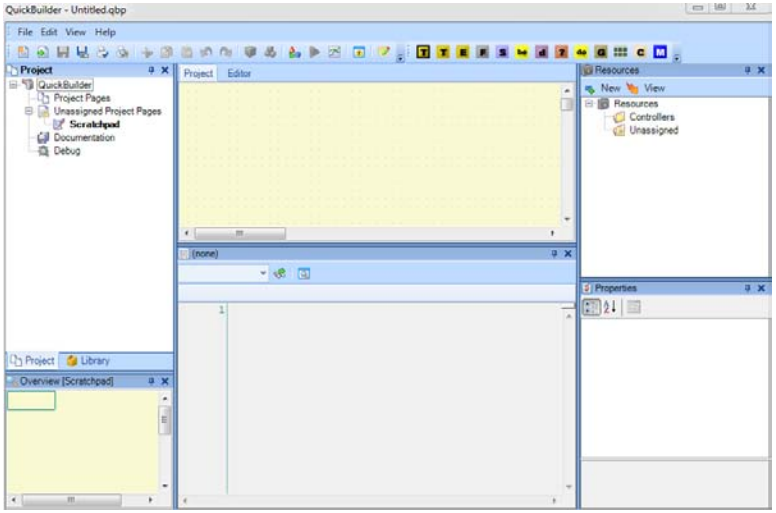
ctc Control Technology Corp.

QuickBuilder

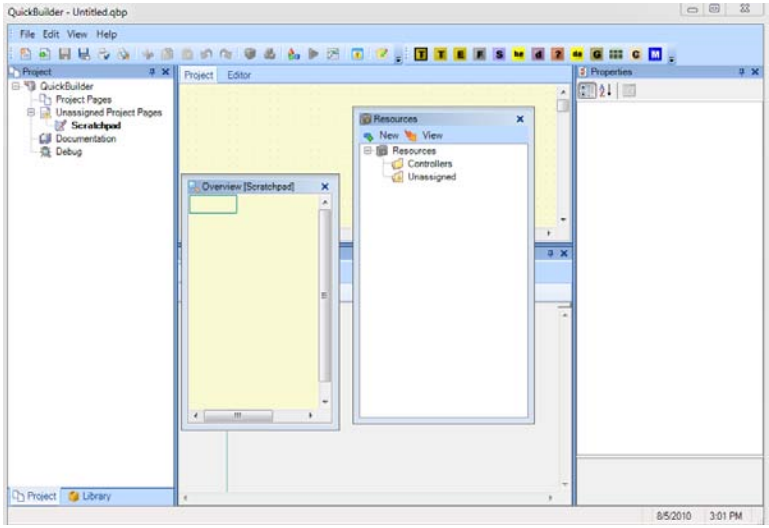
Additional Features

Floating and Dockable Windows

- Allow the user to customize the programming environment
- Environment is saved and reloaded when QuickBuilder is launched



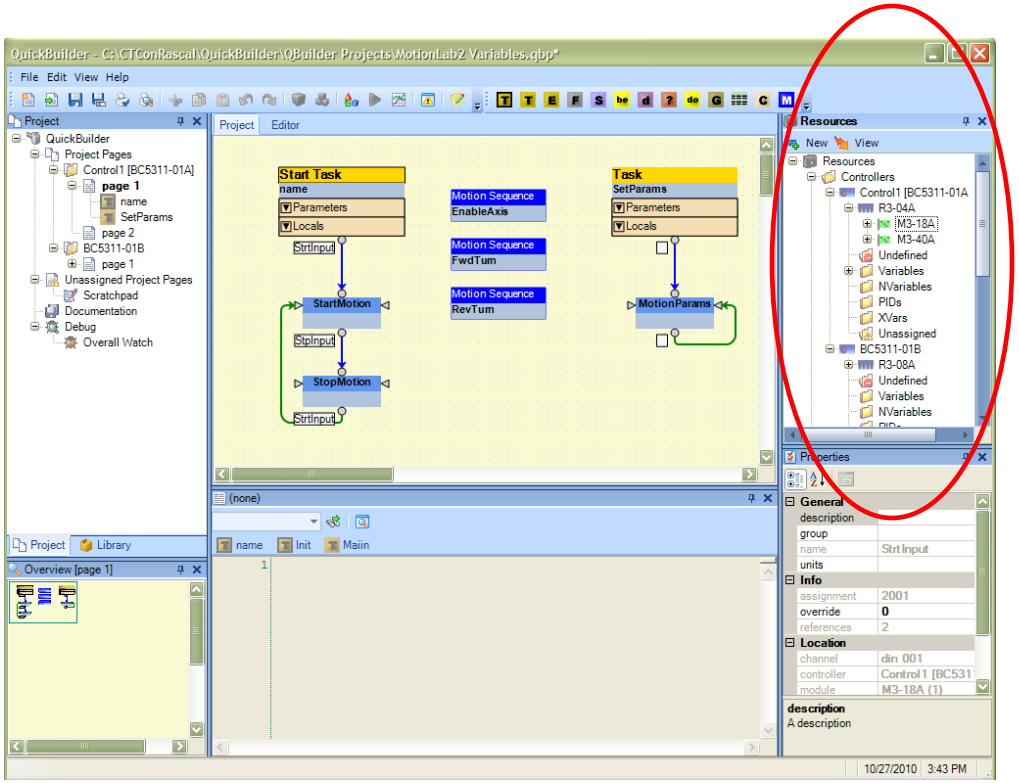
Default Desktop Layout

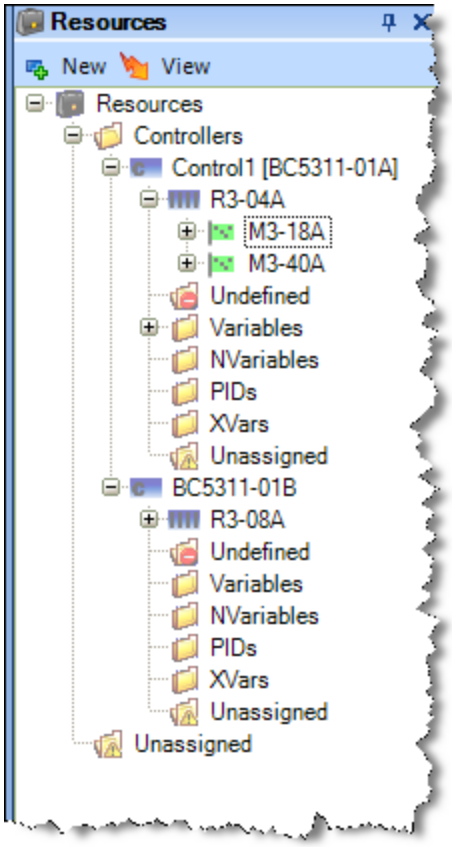


Modified Desktop Layout

ctc Control Technology Corp.

Resource Manager





ctc Control Technology Corp.

Resource Manager Operations

Creating a Resource

- Right click addition of physical resources
- The *New* menu in the Resource Manager
  - Creates *logical* (non-physical) resources

Resource Property Editing

- The Property Inspector allows the user to view and/or edit resource properties

Assigning a Resource

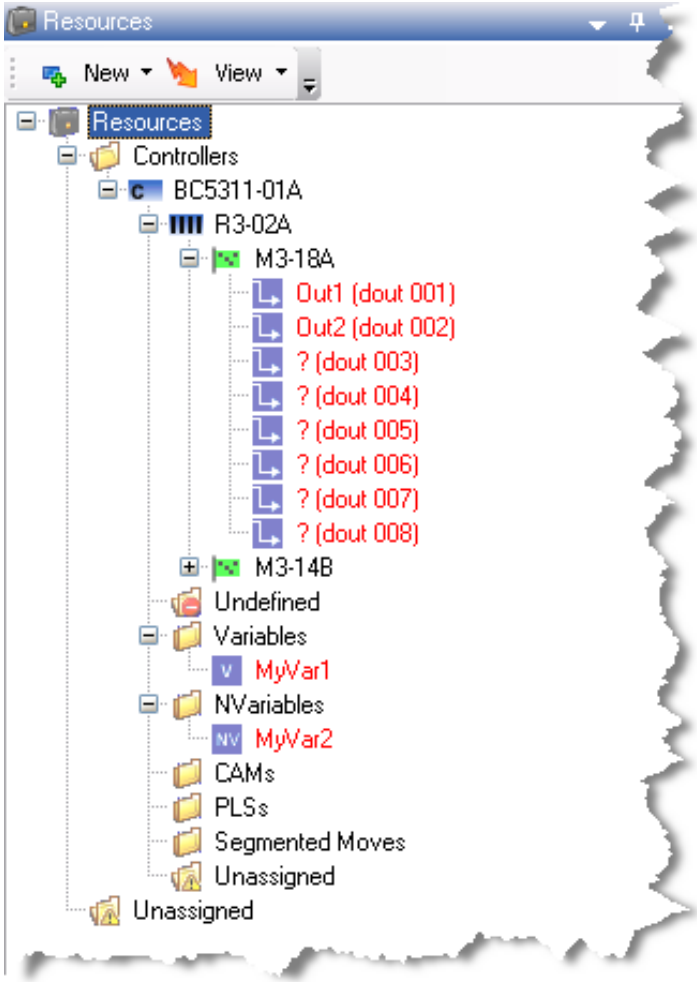
- Used to assign a physical resource to a defined module IO point
- Invoked by right-mouse-clicking (*context-menu)* on a resource

Moving / Exchanging a Resource or Rack

- Used to swap physical resources or entire racks (move-up and move-down)
- Invoked by right-mouse-clicking (*context-menu)* on a resource or rack

Deleting a Resource or Rack

- QuickBuilder prevents the user from deleting *in-use* resources
- Invoked by right-mouse-clicking (*context-menu)* on a resource or Rack



ctc Control Technology Corp.

Resource Manager

Controller Definition

- Racks and I/O
- Non-physical (“logical”) Resources
  - Variables

Undefined Resources

- Placeholders for unknown-type resources
  - When QuickBuilder detects a resource name in QS4 code that it doesn’t recognize, QuickBuilder automatically creates the resource for the user and places the resource in the *Undefined* folder.

Unassigned Resources

- Placeholders for physical and logical resources that have not been assigned

ctc Control Technology Corp.

Resource Types

Physical

- Digital Input
- Digital Output
- Analog Input
- Analog Output
- Axis

ctc Control Technology Corp.

Resource Types

Logical

Variables (Volatile & Non-Volatile)

Data Types

- 32-bit Integer (“int”),
- 64-bit Floating Point (“float”)
- String (“string”)
  - 223-byte maximum length
- Boolean (“boolean”)
- Any (“any”)

Note: Variable names are case sensitive. Spaces and special characters are not allowed.

You may find it useful to routinely use lower-case characters when writing your program, because reserved system variables are typically upper-case and it will help differentiate them.

ctc Control Technology Corp.

Resource Types

Logical

Variables (Volatile & Non-Volatile)

Storage Types

- Scalar (“scalar”)
  - Holds a single value
- One dimensional array (“vector”)
- Two dimensional array (“table”)

Note to QS2 Users: Arrays are used in place data tables. They are much more powerful then data tables also since two-dimensional arrays allow the use of multiple variable types.

Note: Variable names are case sensitive. Spaces and special characters are not allowed.

ctc Control Technology Corp.

Resource Manager Operations

Creating a Resource

- Right click addition of physical resources
- The *New* menu in the Resource Manager
  - Creates *logical* (non-physical) resources

Resource Property Editing

- The Property Inspector allows the user to view and/or edit resource properties

Assigning a Resource

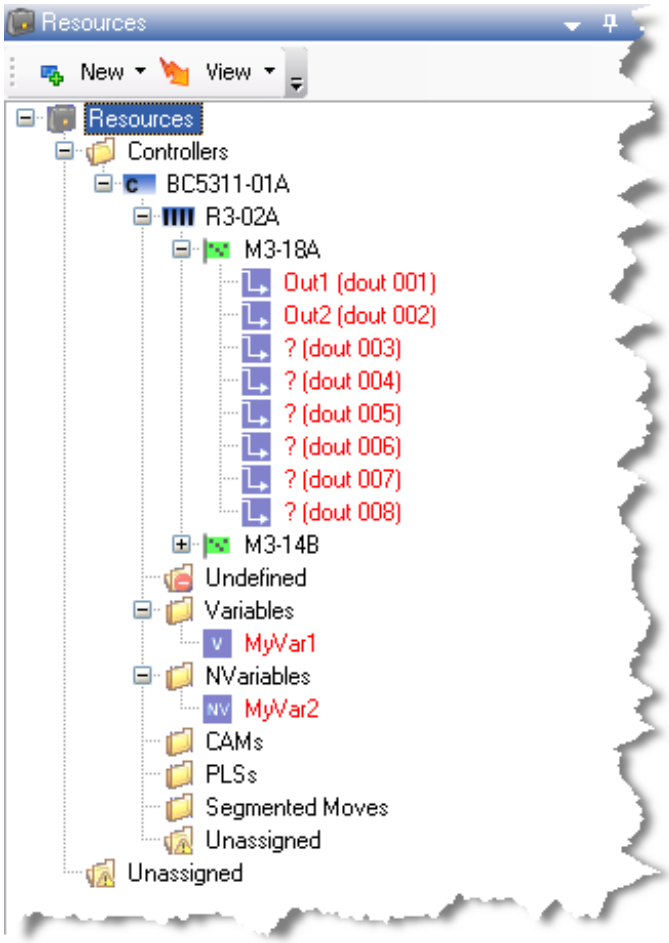
- Used to assign a physical resource to a defined module IO point
- Invoked by right-mouse-clicking (*context-menu)* on a resource

Moving / Exchanging a Resource or Rack

- Used to swap physical resources or entire racks (move-up and move-down)
- Invoked by right-mouse-clicking (*context-menu)* on a resource or rack

Deleting a Resource or Rack

- QuickBuilder prevents the user from deleting *in-use* resources
- Invoked by right-mouse-clicking (*context-menu)* on a resource or Rack



ctc Control Technology Corp.

Resource Manager Limits

485 volatile ints

600 volatile floats, strings or arrays

- Over 250,000 array *cells* holding ints or floats
- Over 40,000 array *cells* holding strings

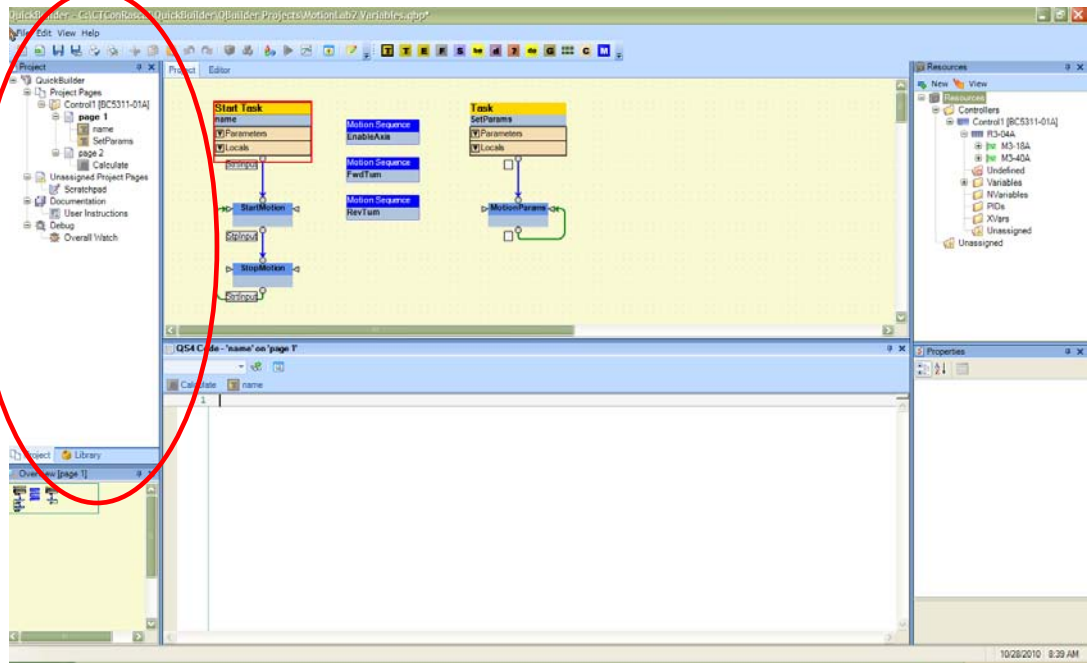
4000 non-volatile ints

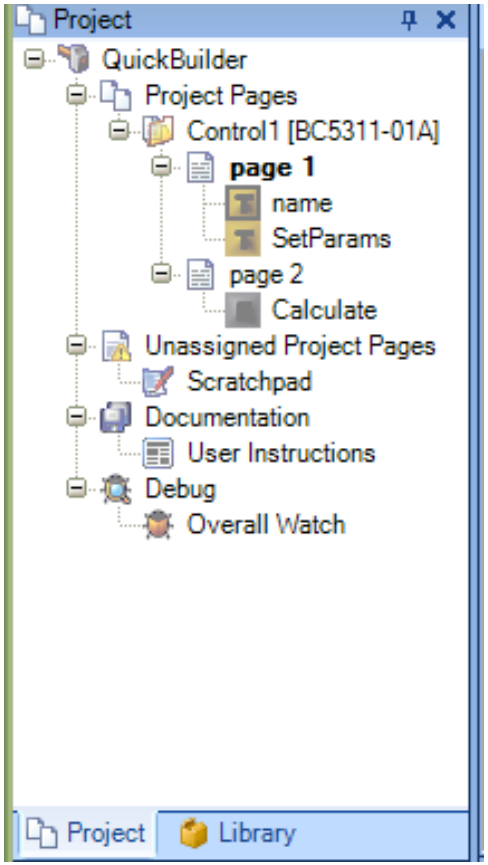
100 non-volatile floats, strings or arrays

- Non-volatile arrays can now be stored to NVRAM. It is suggested you use the save datatable script commands for larger arrays and tables since each element or cell in an array is 256Bytes e.g., a 10 x10 table would occupy 25K of NVRAM space.

ctc Control Technology Corp.

Project Manager





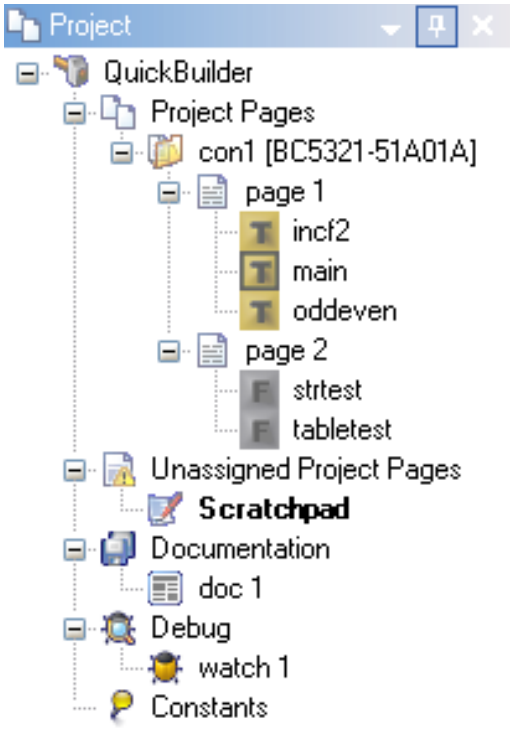
Control Technology Corp.

Project Manager

Controller View

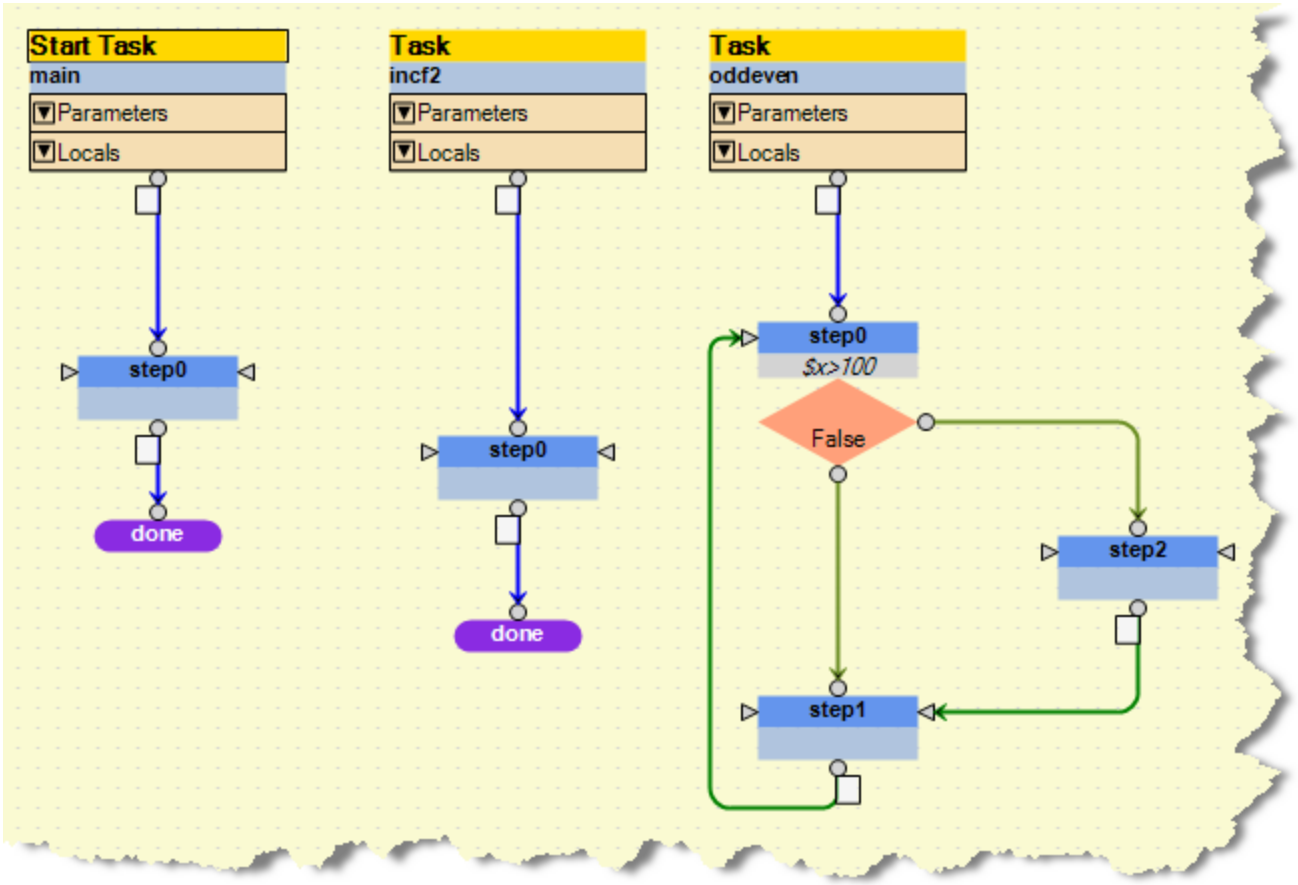
Pages

- Hold the logic for the controller
- Each page is virtually unlimited in size
  - SFC page view shows where printed page breaks occur, similar to Excel.
- Multiple pages allow for logical organization
  - Use of multiple pages does not affect program flow or logic.
  - Use of multiple pages is encouraged to organize the project into manageable sections.



ctc Control Technology Corp.

Page View



ctc
Control Technology Corp.

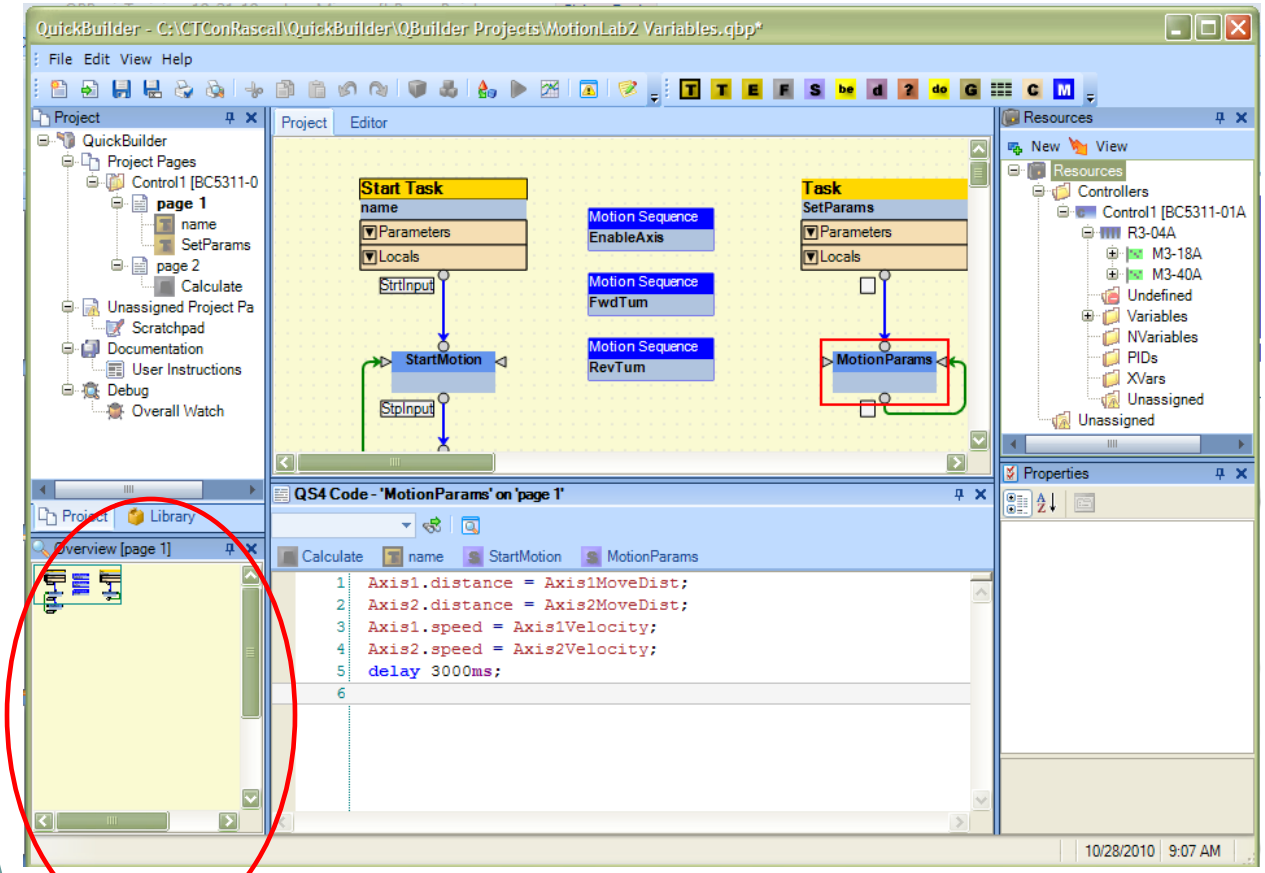
Page View

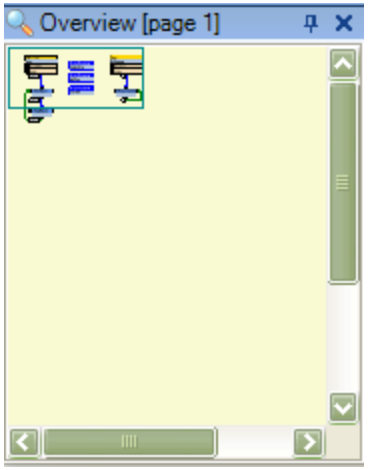
Defines logic and program flow

- Familiar flowchart view
- Simple to edit
  - Toolbar facilitates adding SFC elements to view
  - Simple to connect elements together
    - Auto-layout cleans up connections
  - Typical editing operations such as copy & paste and grouping

ctc Control Technology Corp.

Overview View





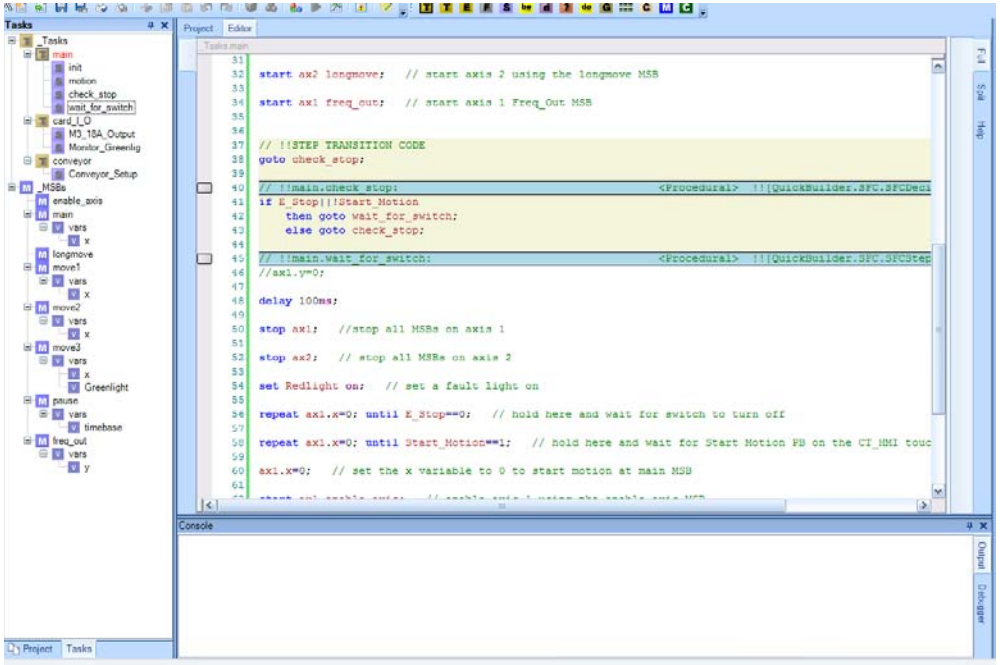
Control Technology Corp.

Editor Window

PROGRAMMING

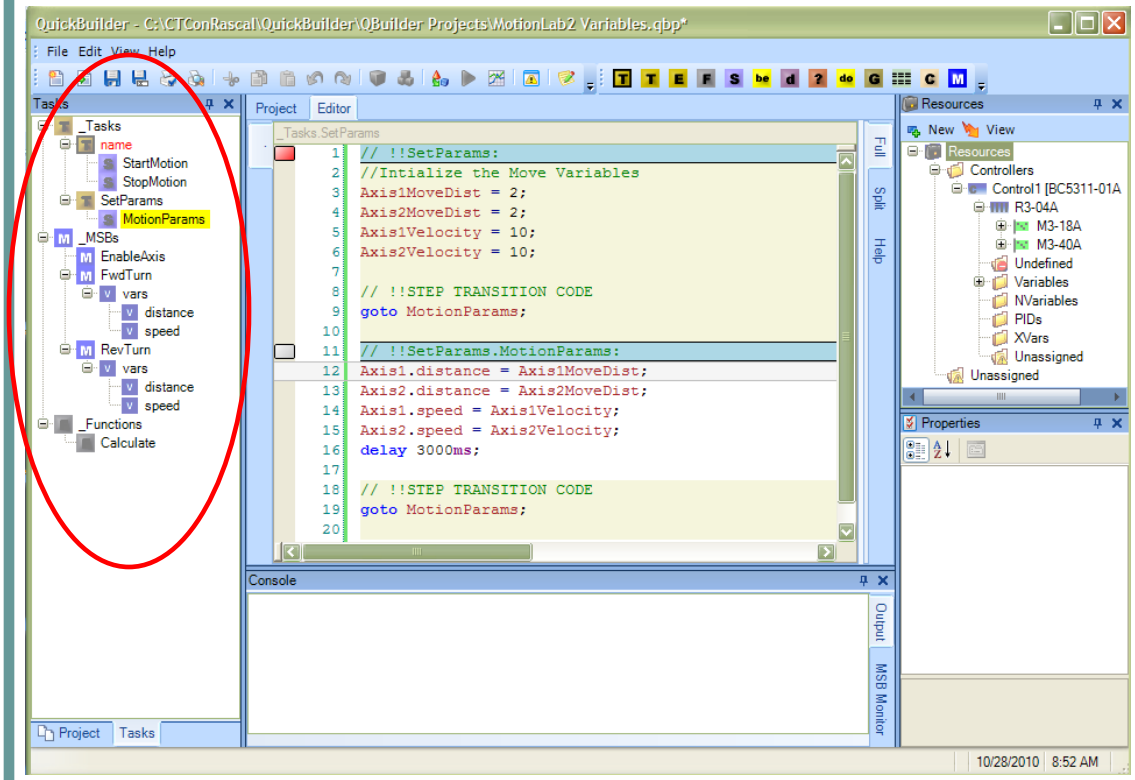
The Editor Window gives you a complete view of all the code within any selected task or MSB.

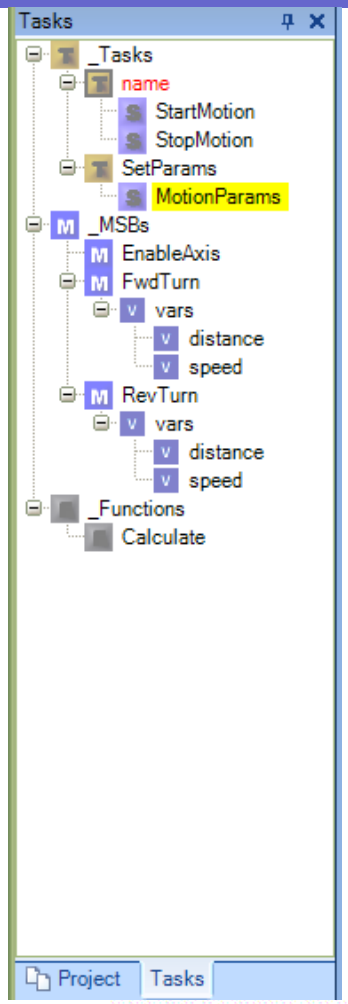
- Split-screen option allows you to look at two tasks or MSBs side by side.
- On demand help screen option gives you help as you code your project.



ctc Control Technology Corp.

Tasks – Editor View

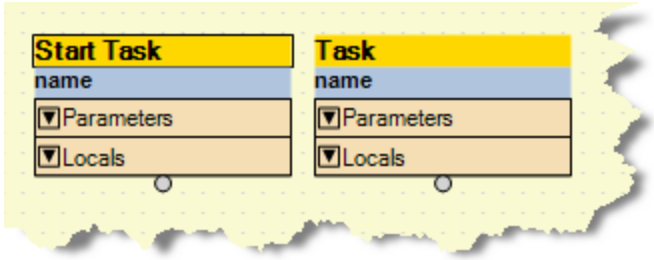




Control Technology Corp.

Project Programming

Tasks



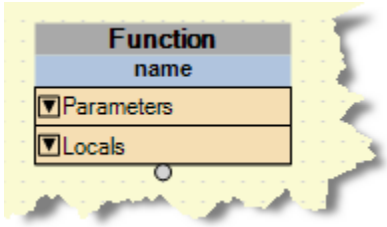
Start Task

- Runs when the program begins
- Only 1 allowed – and 1 is required

Tasks

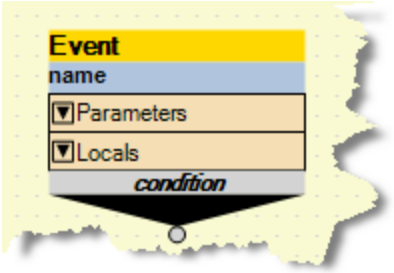
- Need to be started by the Start Task or another running task
- Number of tasks is unlimited, but the maximum number of tasks that can run any one time is 96.

Functions



- Implemented like tasks
- Allows for local variables
- Modular programming

Events



- Implemented like tasks, but wait on a condition
- Automatically re-arm after event-handling concludes
- QS2 “monitor” extension

Transition Conditions

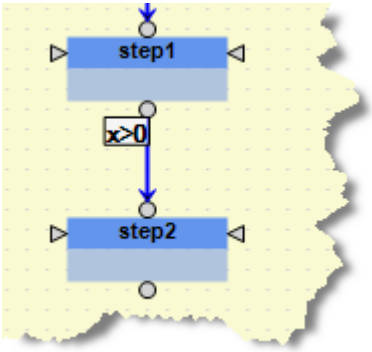
- Useful for creating more readable graphical logic

ctc Control Technology Corp.

Project Programming

Transition Conditions

- Used for more readable graphical logic
- Logic does not proceed unless the condition evaluates to *true so use with caution*
- Empty conditions are *true*

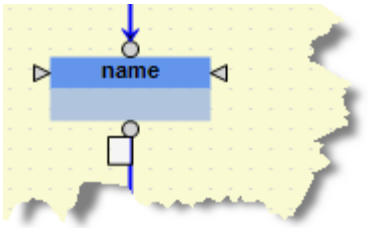


ctc Control Technology Corp.

Project Steps

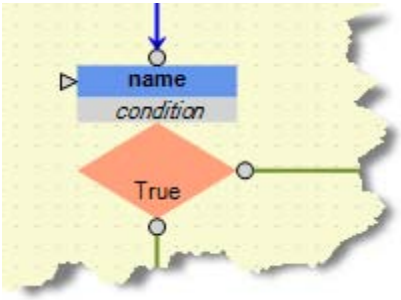
Step

- Holds QS4 code
- Has a transition condition



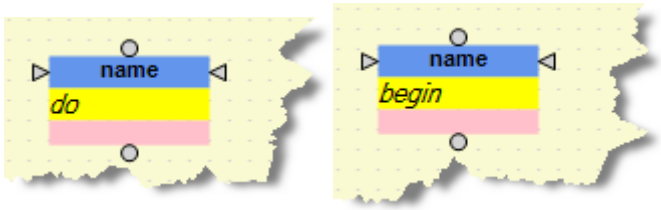
Decision

- Tests for a condition
- Equivalent to QS4 “if” statement
- Only checks once and moves on



Do & Begin

- Starts another task
- Equivalent to QS4 “do”/”begin” statements
  - QS4 statements are more powerful than their graphical counterparts
- Has a transition condition



ctc
Control Technology Corp.

Project Steps

Done

- Ends a Task
- Returns from a Function



Goto

- Graphical version of a QS4 instruction
- Allows graphical program flow
  - No need to connect steps



C-Step

- Allows user-written C-programming

ctc
Control Technology Corp.

Project Editing

Adding an SFC element

- Toolbar
- Auto connection
  - When new elements are inserted with an existing element is selected, the new element is connected to selected element when appropriate to do so

Moving SFC element

- Dragging selected elements move them on the page
- Connections are automatically re-routed

Grouping steps

- Steps can be grouped and named to organize SFC elements
- Sub-grouping is supported

Cut/Copy/Paste

- Control-X cuts one or more SFC elements
- Control-C copies one or more SFC elements
- Control-V pastes previously cut or copied elements

Zoom & Pan

- Mouse scroll-wheel zooms in and out
- Right-mouse drag pans around the view

Control-Drag (Duplicate)

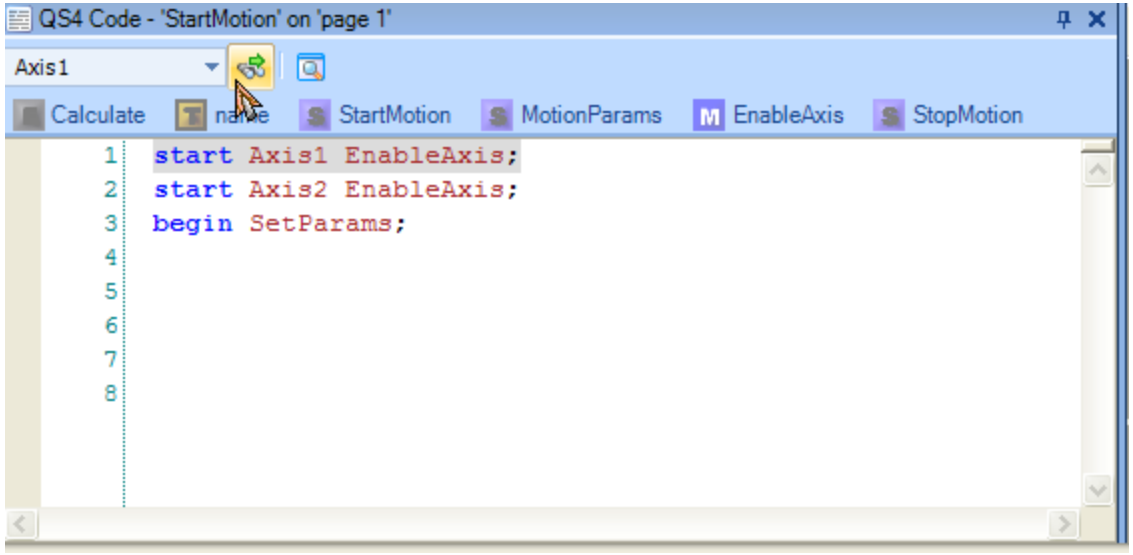
- Holding the control key down while moving (dragging) causes the selected elements to be duplicated

Project Editing

Search

Local Search within a step

- Simply put enter the search info in the search box to find a match and then use F3 to find the next incident

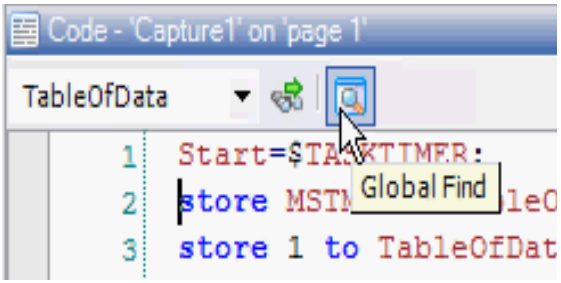


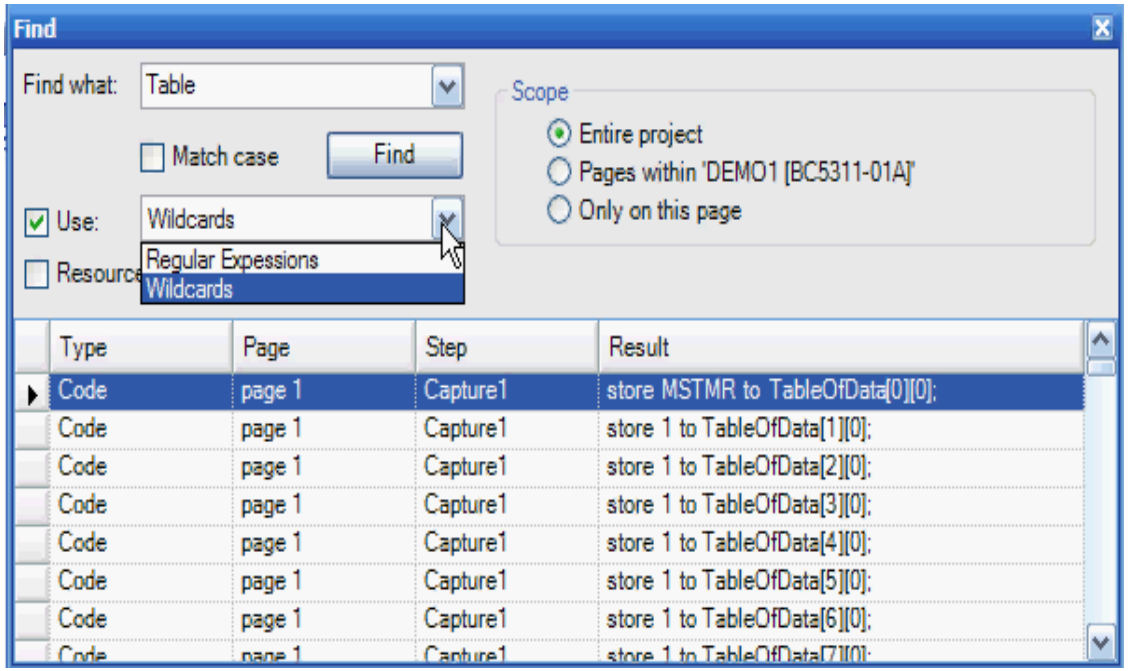
ctc Control Technology Corp.

Project Editing

Search

Global Search



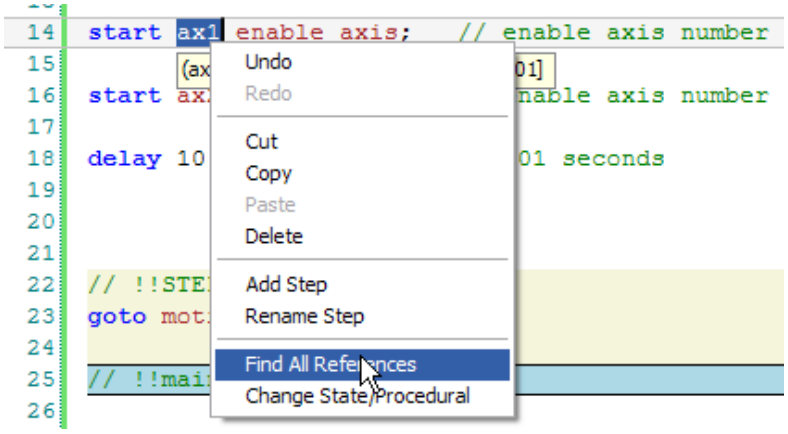


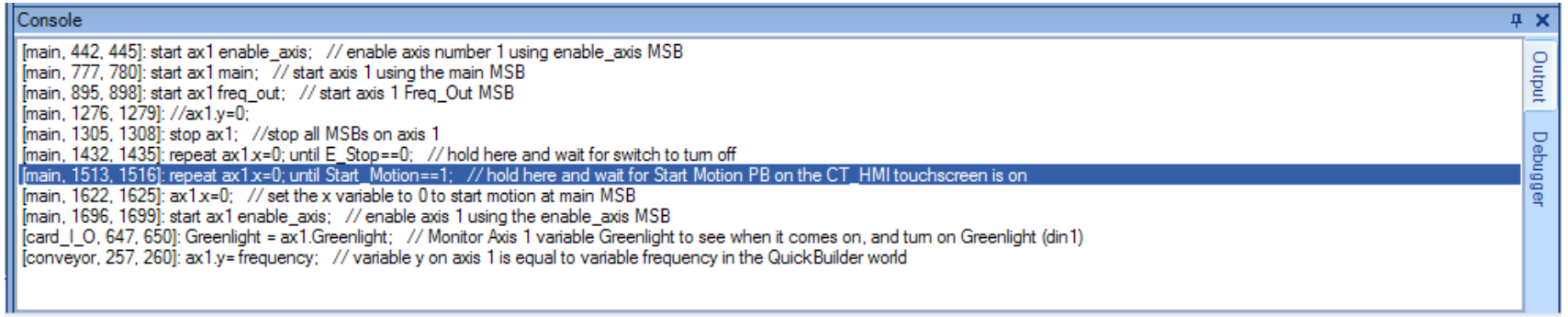
ctc Control Technology Corp.

Editor Search

Find All References

- Highlight anything in the Editor Window.
- Right Click
- Select Find All References
- The Output Tab of the Console will show all references
- Double click any listing and the editor will jump to that reference.





ctc Control Technology Corp.

Editors

Useful Editing Tools

- Undo
  - each step has its own stack to track undo/redo.
- Useful keystrokes
  - Control-M (“*Make* a new SFC step”)
  - F3 (“Find Next”)

ctc Control Technology Corp.

Lab1

The purpose of this Lab is to walk you through the 4 basic steps of creating a QuickBuilder Project to familiarize you with the overall process the QuickBuilder Software itself.

Lab1a CONFIGURATION

- We are going to add a controller, rack, and I/O modules for our required configuration.

Lab1b PROGRAMMING

- We are going to define overall logic in SFC graphical form and code the required steps using Quickstep 4 code.

ctc Control Technology Corp.

Lab1

Lab1c COMMISSIONING

- We will translate, correct errors (as required) and then publish the project for execution on the controller
- We will access the status/control window and a watch window to confirm/debug the project.

ctc Control Technology Corp.

Lab1

Lab1d MAINTAINING

- We will auto generate project documentation including cross reference tables. Normally this is when we would add additional documentation using the internal word processor, or by attaching external files, such as spreadsheets, CAD files, or photos.

- Maintaining also involves monitoring the controller locally or remotely, and making updates or changes to an active project over a local or secure remote network connection.

Resources - System Variables

System Variables

- System variables are pre-defined named resources that the QS4 programmer can access
  - Where QS2 had special registers such as 13002 (the millisecond timer), QS4 uses names such as $TASKTIMER to refer to such items
  - QS4 system variables are more powerful since they opens up more possibilities – such as the $TASKTIMER variable which is *local* and *unique* to a Task

ctc Control Technology Corp.

Resources - System Variables

Accessed like local variables (“$”)

- **$TASKTIMER**
  - Increments every millisecond; set to zero when task begins
  - Writeable
- **$DINPUTS**
  - An array of all the digital inputs
  - 1-based (**$DINPUTS[1]** is the first digital input)
- **$DOUTPUTS**
  - An array of all the digital outputs
  - Writeable
  - 1-based (**$DOUTPUTS[1]** is the first digital output)
- **$REGISTERS**
  - An array of all the QS2-oriented register
  - Writeable
  - **$REGISTERS[13002]** is the millisecond timer
- **$TRIGGER**
  - Setting this to one causes QuickScope to start a waiting Capture.

ctc Control Technology Corp.

Numeric Expressions

Numeric Expressions are composed of:

- Variables
  - 32-bit Integer (“int”),
  - 64-bit Floating Point (“float”)
  - String (“string”)
    - 223-byte maximum length
  - Boolean (“boolean”)
  - Any (“any”)
  - Array indexes
    - `variable[index]` accesses a 1-dimensional *vector*
    - `variable[index][index]` accesses a 2-dimensional *table*
- Constants (must be all lower case or all upper case)
  - pi PI
  - `on ON off OFF true TRUE false FALSE`

ctc Control Technology Corp.

Numeric Expressions

Numeric Expressions are composed of:

- Operators
  - `+ - * /` arithmetic
  - `%` modulo
  - `| & ^ >> << ~` bit operators
  - `&& || !` logical operators
  - `> >= < <= == !=` comparison operators

  Note: that a comparison uses == rather then the = used for the purpose of assigning a numeric expression.
- Parentheses
- Built-in Arithmetic Functions and String Functions

ctc Control Technology Corp.

Numeric Expressions

Bit Operators

- & - bit-wise *and*
  - Logically *ands each* bit in the left-hand side with its corresponding bit in the right-hand side
    - 14 (1110) & 5 (0101) = 4 (0100)
- | - bit-wise *or*
  - Logically *ors each* bit in the left-hand side with its corresponding bit in the right-hand side
    - 14 (1110) | 5 (0101) = 15 (1111)
- ^ - bit-wise *exclusive or*
  - Logically *exclusive-ors each* bit in the left-hand side with its corresponding bit in the right-hand side
    - *Exclusive-or* is an *or* with the caveat that both bits cannot be on
    - 14 (1110) ^ 5 (0101) = 11 (1011)

ctc Control Technology Corp.

Numeric Expressions

Bit Operators

- >> - shift-right
  - **(x >> n)** Arithmetic shift right of *x* by *n* bits
    - Introduces a 0 for the empty bit on the left
    - 14 (1110) >> 2 = 3 (0011)
    - Effectively a divide by 2^n
- << - shift-left
  - **(x << n)** Arithmetic shift left of *x* by *n* bits
    - Introduces a 0 for the empty bit on the right
    - 3 (0011) << 2 = 12 (1100)
    - Effectively a multiply by 2^n
- ~ - bit-wise *not*
  - Operates on all bits
    - ~ 0011 = 1111 1111 1111 1111 1111 1111 1111 1100

ctc Control Technology Corp.

Numeric Expressions

Logical Operators

- && - logical *and*
  - Treats the right-hand side and the left-hand side as Booleans (any non-zero value is *true*) and performs an *and* returning a Boolean
    - 14 (1110) && 5 (0101) = 1 (0001)
    - 14 (1110) && 0 (0000) = 0 (0000)
    - StartInput && (Speed > 10)
- || - logical *or*
  - Treats the right-hand side and the left-hand-side as Booleans (any non-zero value is *true*) and performs an *or* returning a Boolean
    - 14 (1110) || 5 (0101) = 1 (0001)
    - 14 (1110) || 0 (0000) = 1 (0001)
    - Estop || (PositionErr > 5000)
- ! – logical *not*
  - Treats the right-hand side as a Boolean (any non-zero value is *true*) and then performs a *not* returning a Boolean
    - !14 = 0
    - !1 = 0
    - !0 = 1
    - !StartInput
- Comparison operators are essentially logical operators as they return Booleans

ctc Control Technology Corp.

Numeric Expressions

Logical Expressions

- Examples
  - `Output1 = Input3 && Input4 && (Xaxis_position < 0);`
    - **Output1** is set to true when **Input3** and **Input4** are both on *and* when **Xaxis_position** is less than 0
  - `Fail = !Sensor1 || !Sensor2 || !Sensor3 || !Sensor4;`
    - **Fail** is set to true when *any* of the four sensor inputs are false
  - `JogForw = JogFInput && !JogRInput;`
    - **JogForw** is set to true when the **JogFInput** is true and the **JogRInput** is false
  - `JogRev = !JogFInput && JogRInput;`
    - **JogRev** is set to true when the **JogFInput** is false and the **JogRInput** is true

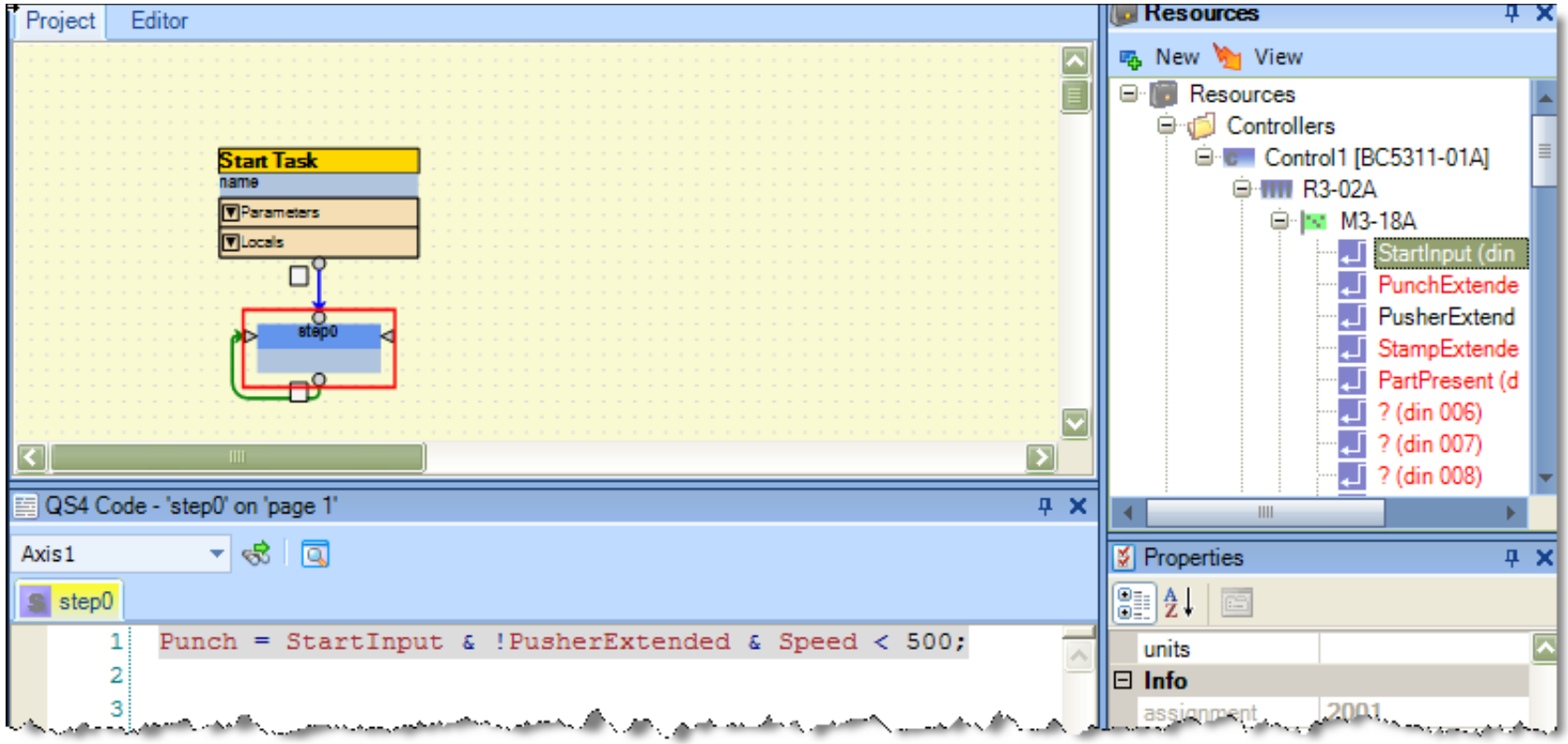
ctc Control Technology Corp.

Quick Lab 1 - Expressions

1. Open Lab1a, since it's configured for your controller. Save it as QuickLab1.
2. Write a program that turns on output 1 if input 1 is on, 3 is off, and an integer variable named speed is < 500.
   Use an expression to accomplish this.
3. Commission this program.
4. Resave it as QuickLab1.

ctc Control Technology Corp.

Quick Lab 1 - Expressions



ctc Control Technology Corp.

String Expressions

String Expressions

- **string(…)**
  - `x = string("Hello There");`
- Variables
  - Array Indexing is supported for strings
    - `x[0] = string("Hello There");`
    - `x[1] = string("Goodbye");`
- Constants
  - Enclosed in double-quotes
  - C-notation for special characters
    - `x = string("Hello\rThere");`
      - x is ***Hello*** followed by a **carriage return** then ***There***
- Concatenation Operator ("+")
  - `Y = string("Joe");`
  - `x = string("Hello There" + y);`
- Built-in String Functions
  - Familiar VB functions such as **left**, **right** and **mid**

ctc Control Technology Corp.

Step Programming

Overview

Comments

- Single-line
  - `// a single line comment starts with two forward slashes`
- Multi-line
  - ```
    /*
       Multi-line comments start with a forward slash and star
       and end with a star and forward slash
    */
    ```

Grouping Code

- ```
  ## group name
     statement
     statement
     statement
  ##.
  ```

ctc Control Technology Corp.

Step Programming

Overview

Free-form input

- Spaces, tabs and line-breaks are ignored
- The following are all equivalent :

```
x=10;
x = 10;
x   =      10  ;
```

Semicolons (“end of statement”)

- All statements must end with a semicolon
- Required since QS4 is free-form

Error Indicators

- Red-squiggle underline shows error
- Error indicators are live and update as the user types
- Note that incomplete commands will show up as having a syntax error
  - The two examples show an incomplete and then complete command

```
Punch = StartInput & !PusherExtended & Speed < 500
```

```
Punch = StartInput & !PusherExtended & Speed < 500;
```

ctc Control Technology Corp.

Step Programming

Overview

Indentation

- QS4 code editor automatically indents code as appropriate

Curly-Brackets (“block statement”)

- Allows for multi-line statements
- Code contained in curly brackets is referred to as a ***block*** statement
- A block statement contains multiple statements, yet is equivalent to a single statement because those statements are encapsulated in curly-brackets

ctc
Control Technology Corp.

Step Programming

Block Statement

used to organize code and within other statements such as “if then else”, “while”, “repeat” and “for” statements.

```
repeat {
    set Alarm on;
    delay 500ms;
    set Alarm off;
    delay 500ms;
    } until ResetInput;
```

ctc Control Technology Corp.

Step Programming

Assignment

- *variable* = expression;
- *variable* += expression; (i.e. count+=1; same as count=count+1;)
- *variable* -= expression; (count-=1; same as count = count -1)
- **store** *expression* **to** *location (QS2 equivalent)*;

```
/* If variable count is greater then 10 goto step 2 otherwise
   increment count and goto step0 */
if count >10 then
goto step2;
else {
count+=1;          // increment count by 1
goto step0;}
```

ctc Control Technology Corp.

Step Programming

Set

- **set** *output list* **on | off | ON | OFF**;

```
set Motor on;                    // Turn on a Motor Starter Output

set FastJogFlag1 off;            // Turn off a flag/bit

set Motor1, Motor2 off;          // Turn off two Motor outputs
```

ctc Control Technology Corp.

Step Programming

Set Bit / Clear Bit

- **setbit BIT***n variable*;
- **setbit** *n variable*;
- **clrbit BIT***n variable*;
- **clrbit** *n variable*;

setbit BIT7 x;	// sets bit7 of the x variable
setbit 5 x;	// sets bit5 of the x variable
clrbit BIT10 x;	// clears bit 10 of the x variable
y = bit(x,7);	// y is true (1) when bit7 of x is 1/true

ctc Control Technology Corp.

Step Programming

Goto

- **goto** *destination*;

```
goto StartMotor;   /*tells the controller to proceed to the
                     StartMotor step */
goto next;         /*tells the controller to proceed to whatever the
                   next step in the sequence is*/
```

ctc Control Technology Corp.

Step Programming

Delay

The ***delay*** instruction is similar to the step available in QS2. This instruction causes the controller to proceed after a specified amount of time has elapsed. This time delay is specified as:

- **delay** *expression* **ms**;

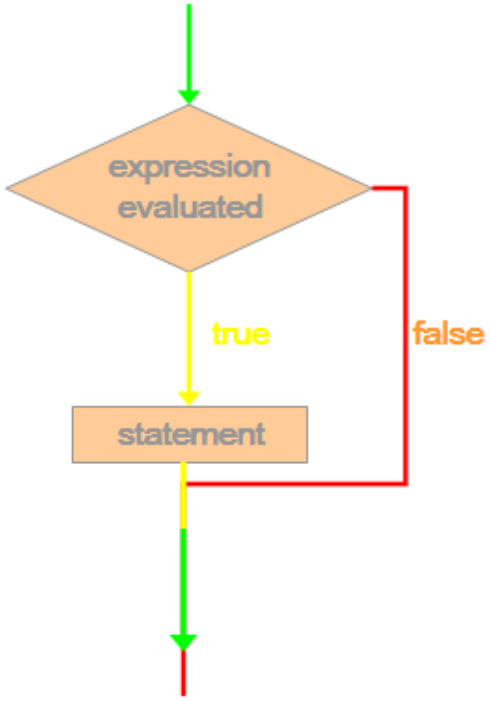
```
set Ejector on;          // Turn on the Ejector Output
delay 1000 ms;           // Delay 1 second
set Ejector off;         // Turn off the Ejector Output
```

Note: the internal resolution of the controller is 1 millisecond, and all times will be rounded appropriately.

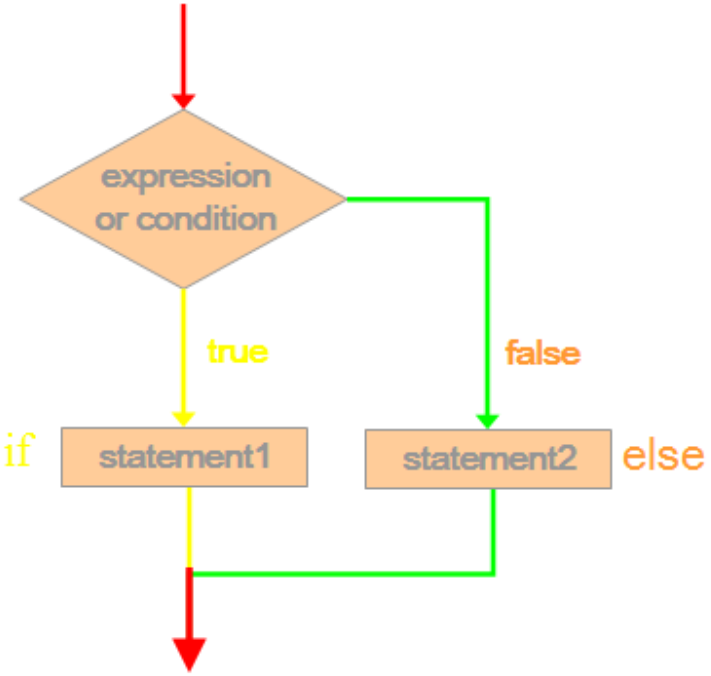
ctc Control Technology Corp.

Step Programming

If / Then / Else




```
if ( condition )
    statement1;
else
    statement2;
```



ctc
Control Technology Corp.

Step Programming

If / Then / Else

1. **if** *expression* **goto** *step*;
2. **if** *expression* **then** *statement;*
3. **if** *expression* **then** *statement;* **else** *statement;*
4. **if** *expression* **then goto** *step;* **else goto** *step;*

```
if x >= 8 then goto ProcessComplete; else goto next;
```

```
if dig1 then {              // if dig1 is true
      set digout1 off;      // set digout1 off
      x = 0;                // and make x equal to 0
      }
else x = 1;                 // if dig1 is not true make x equal to 1
```

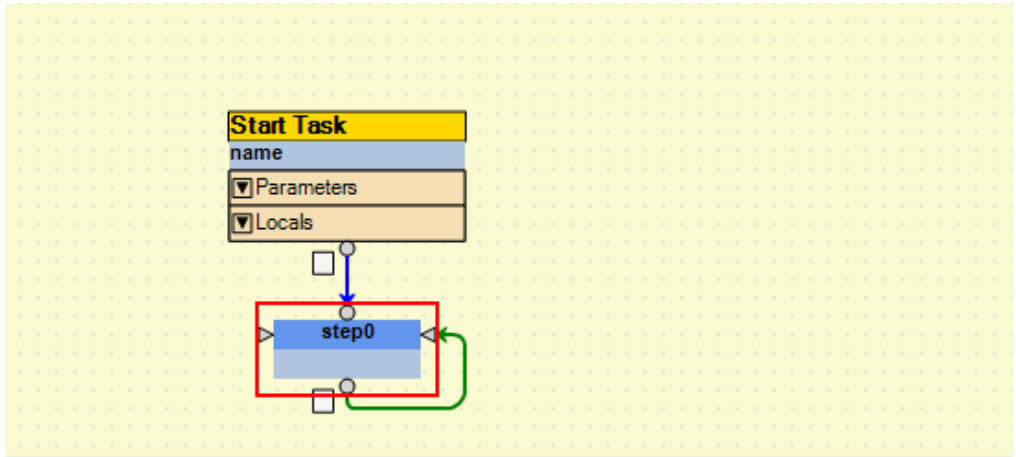
ctc Control Technology Corp.

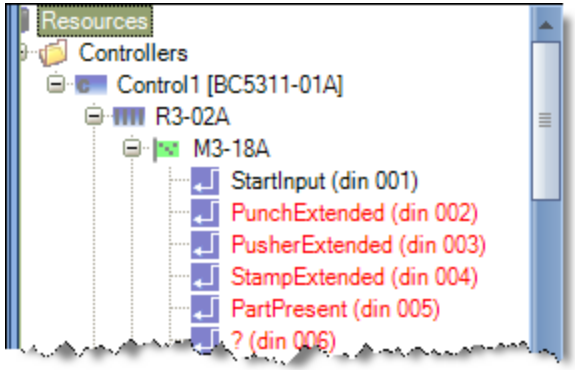
Quick Lab 2 – if and set

1. Open Lab1a, since it's configured for your controller. Save it as QuickLab2.
2. Write a program that turns on outputs 1, 3 and 5 if input 1 is on and turns them off if it is not on. *Use an if statement and set to accomplish this.*
3. Commission this program.
4. Resave it as QuickLab2.

ctc Control Technology Corp.

Quick Lab 2 – if and set





QS4 Code - 'step0' on 'page 1'

name step0

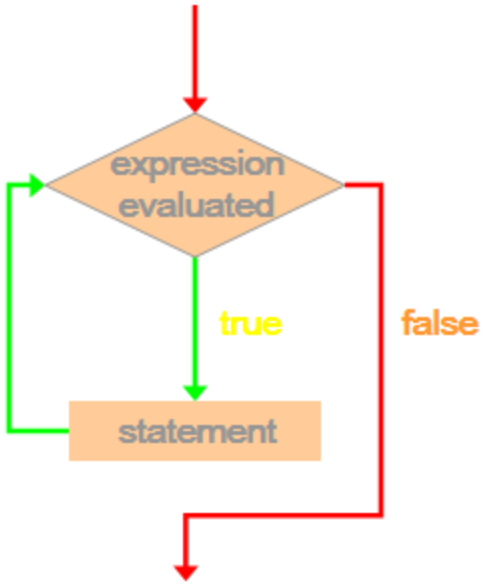
```
1  /*If StartInput is on turn on Punch, Pusher, Indexer
2  else if it's off turn them off */
3
4  if StartInput then
5      set Punch, Stamp, Indexer on;
6  else
7      set Punch, Stamp, Indexer off;
8  //repeat this step
9
```

ctc Control Technology Corp.

Step Programming

While

- **while** *expression*



ctc
Control Technology Corp.

Step Programming

While

- **while** *expression* **repeat** *statement*;

```
while BatchCount <= 10 repeat {  //while BatchCount is less then 10
      call BatchRun;             //Call the Function BatchRun
      BatchCount+=1;             // Increment BatchCount by 1
}
```

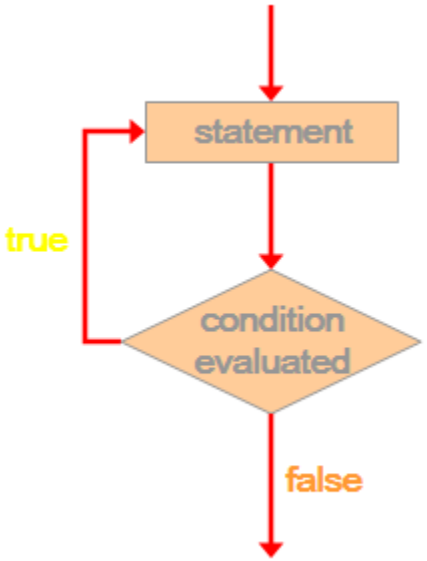
```
while !StartIn repeat {          /*while StartIn is is not true(off)
}                                Do nothing (just wait)
      goto ThisStep;             Goto the beginning of this step*/
```

ctc Control Technology Corp.

Step Programming

Repeat / Until

- **repeat** *statement*; **until** *expression or condition*;



With a repeat loop the statement is always executed at least once.

ctc Control Technology Corp.

Step Programming

Repeat / Until

- **repeat** *statement*; **until** *expression*;

```
repeat {                        // Repeat
Call BatchRun;                  // Call the Function BatchRun
BatchCount+=1;                  // Increment BatchCount by 1
}
until BatchCount >= 10          /* move on After BatchCount is greater then or
                                   equal to 10 */
```

```
repeat {                        // Repeat
}                               // Do nothing (just wait)
until StartIn;                  // while StartIn true (on)
goto ThisStep;                  // Goto the beginning of this step
```

Notice that Repeat / Until works like the While statement only in reverse. The condition is checked after the statements are run, so a repeat statement will always execute once.

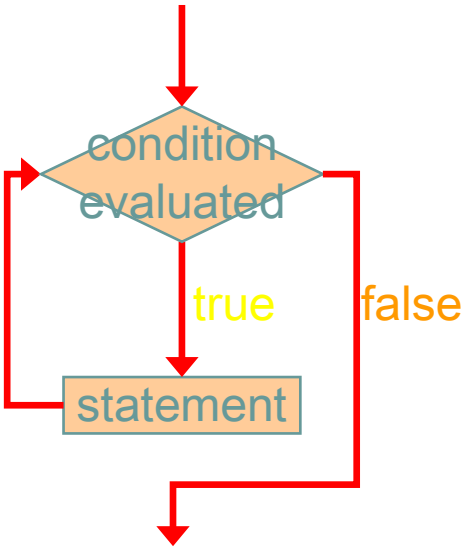
ctc Control Technology Corp.

Step Programming

Comparing while and repeat/until

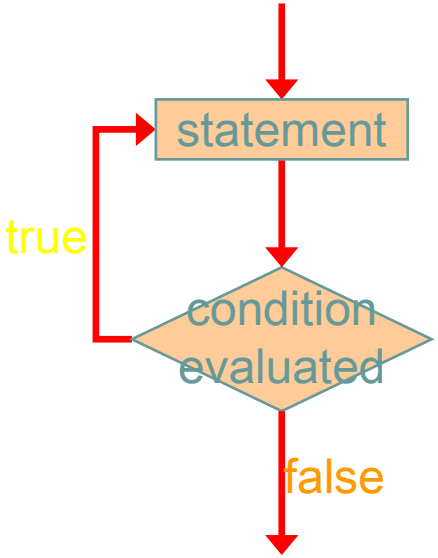
executes 0 or more times

while loop



repeat loop

executes at least once



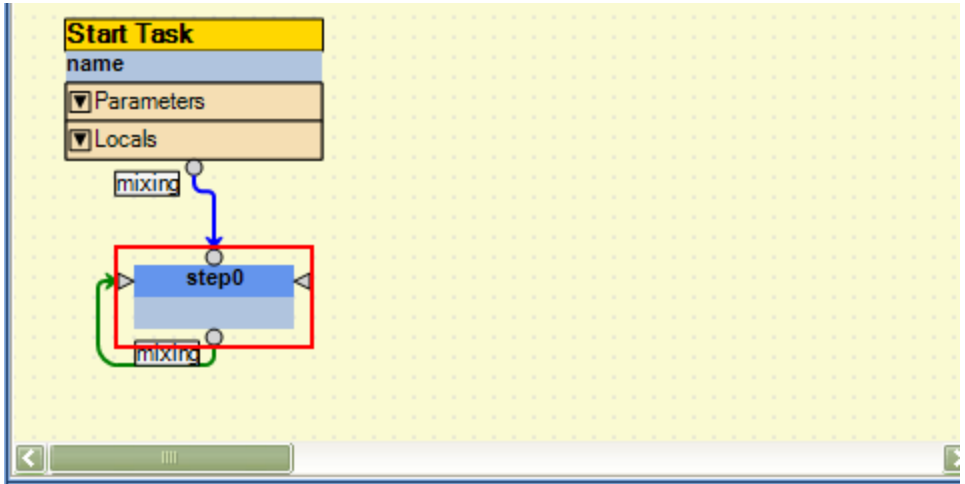
ctc Control Technology Corp.

Quick Lab 3 – while&delay

1. We are now going to create a simple program using a while/repeat.
2. Write a program that turns on output 1, delays 500ms, and then turns it off for 500ms while a “mixing” variable is on and the $TASKTIMER is < 10 seconds.
3. Commission this program.
4. Resave it as QuickLab3.

ctc Control Technology Corp.

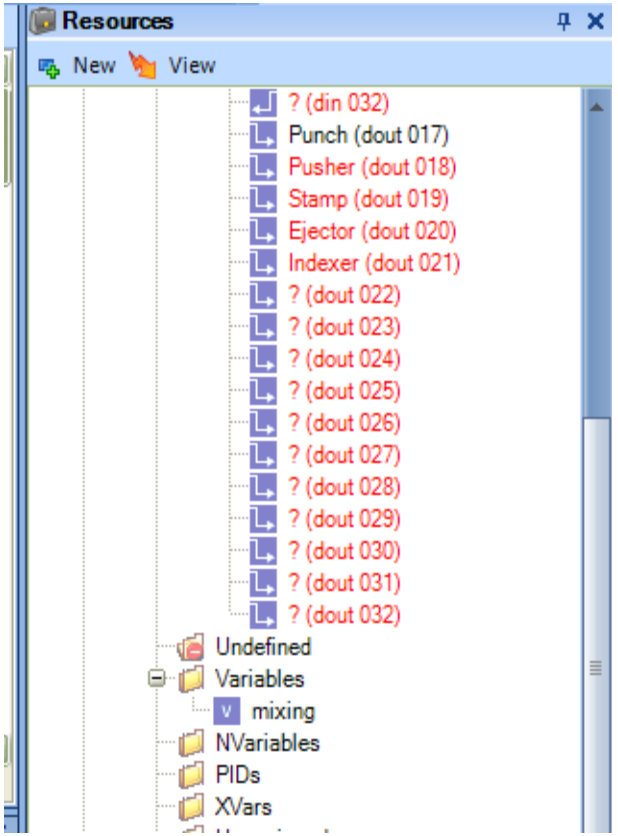
Quick Lab 3 – while&delay



QS4 Code - 'step0' on 'page 1'

step0 name

```
1  $TASKTIMER = 0;
2  while mixing && $TASKTIMER <= 10000 repeat
3      {set Punch on;
4      delay 500ms;
5      set Punch off;
6      delay 500 ms;}
7  mixing = 0;
8
```



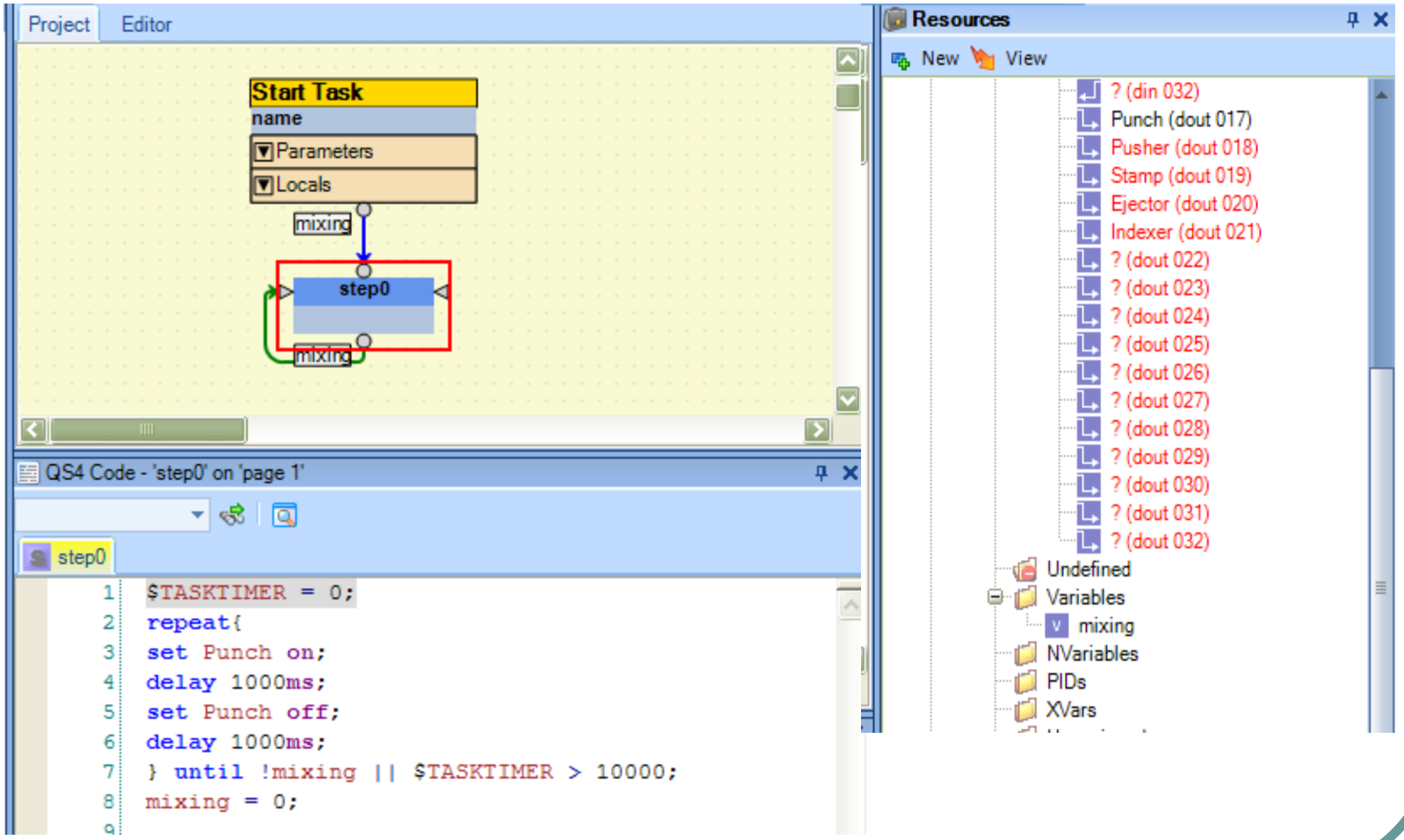
ctc Control Technology Corp.

Quick Lab 4 – repeat&delay

1. We are now going to create a simple program using a repeat/until.
2. Write a program that waits for mixing to be true, turns on output 1, delays 500ms, and then turns it off for 500ms and repeat this until a “mixing” variable is off or the $TASKTIMER is > 10 seconds.
3. Commission this program.
4. Resave it as QuickLab4.

ctc Control Technology Corp.

Quick Lab 4 – repeat&delay

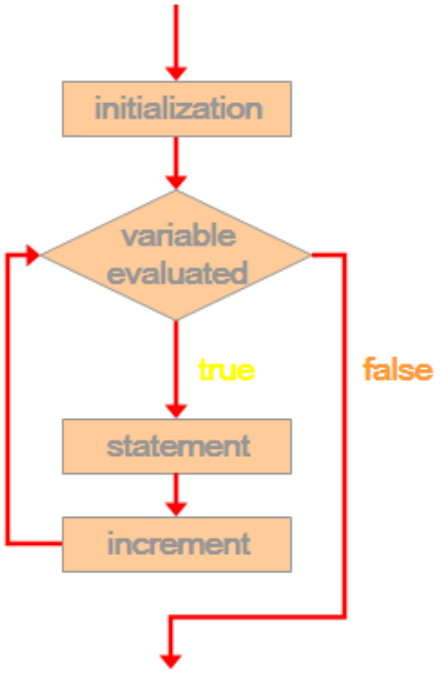


ctc Control Technology Corp.

Step Programming

For

- **for** *variable* = *start* **to** *end* **repeat** *statement*;
- **for** *variable* = *start* **to** *end* **by** *increment* **repeat** *statement*;



ctc Control Technology Corp.

Step Programming

For

- **for** *variable* = *start* **to** *end* **repeat** *statement*;
- **for** *variable* = *start* **to** *end* **by** *increment* **repeat** *statement*;

```
for i = 1 to 10 repeat          //Repeat this loop 10 x
{
Call BatchRun;                  // Call the Function BatchRun
}
```

To iterate the variable "i" from 0 to 6 (inclusive) by steps of 2 (0, 2, 4, 6):

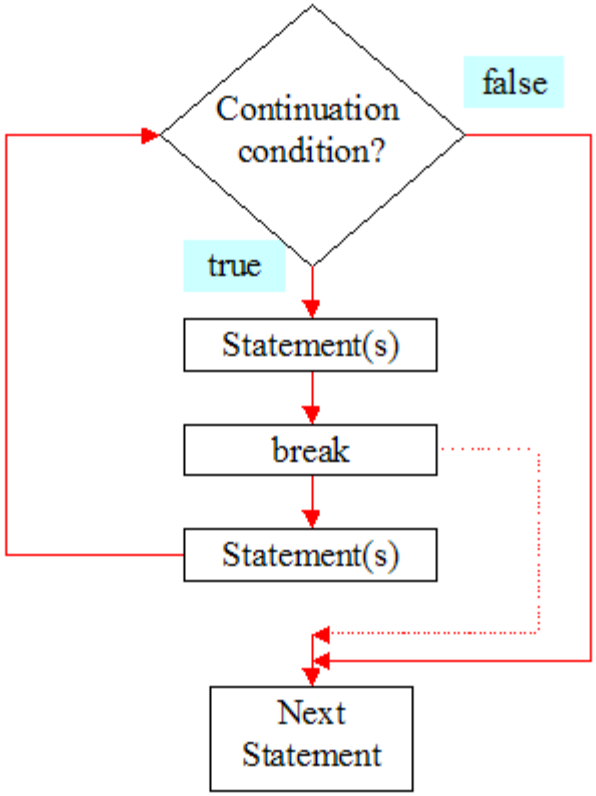
```
for i = 0 to 6 by 2 repeat
{
statement;                      // These statements would be executed 4x
Pos = i  +  2.5;                // Note that i will be 2.5, 4.5, 6.5, and then 8.5.
}
```

ctc Control Technology Corp.

Step Programming

Break

The ***break*** instruction aborts a ***while***, ***for*** or ***repeat*** instruction. When this instruction is encountered, no more looping or iterating occurs and the "exit" condition (such as the *while's until* condition) is not tested.



ctc Control Technology Corp.

Step Programming

Break

The ***break*** instruction aborts a ***while***, ***for*** or ***repeat*** instruction. When this instruction is encountered, no more looping or iterating occurs and the "exit" condition (such as the *while's until* condition) is not tested.

- **break**;

```
for i = 1 to 100 repeat
{
statement1;
...
if (x1speed > 100) then break;
...
}
```

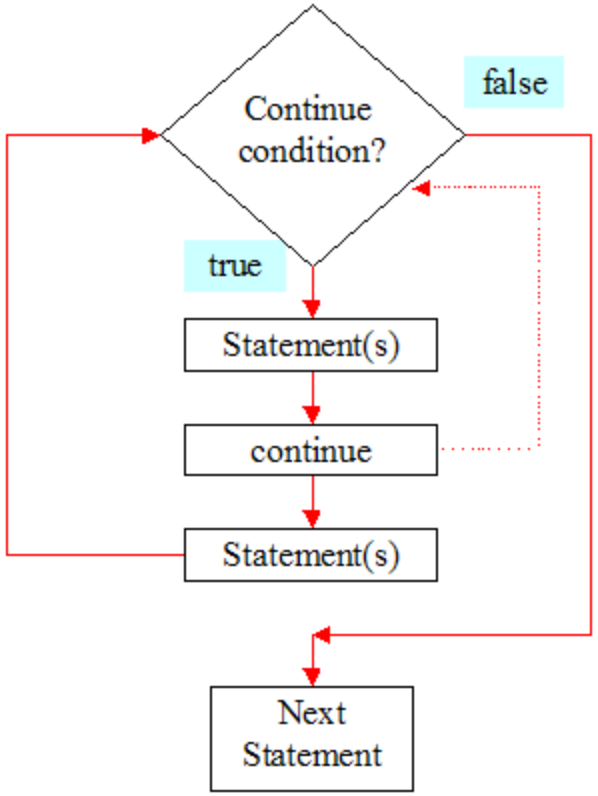
```
while digin1 repeat {
statement1;
statement2;
if MotorError then break;
...
}
statement3;
```

ctc Control Technology Corp.

Step Programming

Continue

The ***continue*** instruction immediately begins the next iteration ("loop") of a ***while***, ***for*** or ***repeat*** instruction. When this instruction is encountered, no more statements are executed until the iteration condition is tested (*while* and *repeat*) or the next iteration begins (*for*).



ctc Control Technology Corp.

Step Programming

Continue

The ***continue*** instruction immediately begins the next iteration ("loop") of a ***while***, ***for*** or ***repeat*** instruction. When this instruction is encountered, no more statements are executed until the iteration condition is tested (*while* and *repeat*) or the next iteration begins (*for*).

- **continue**;

```
for i = 1 to 100 repeat
{
...
statement9;
if (x1speed > 100) then continue;
statement10;
}
```

```
while digin1 repeat {
statement1;
statement2;
if Estop == -1 then continue;
statement3;
}
```

ctc Control Technology Corp.

Step Programming

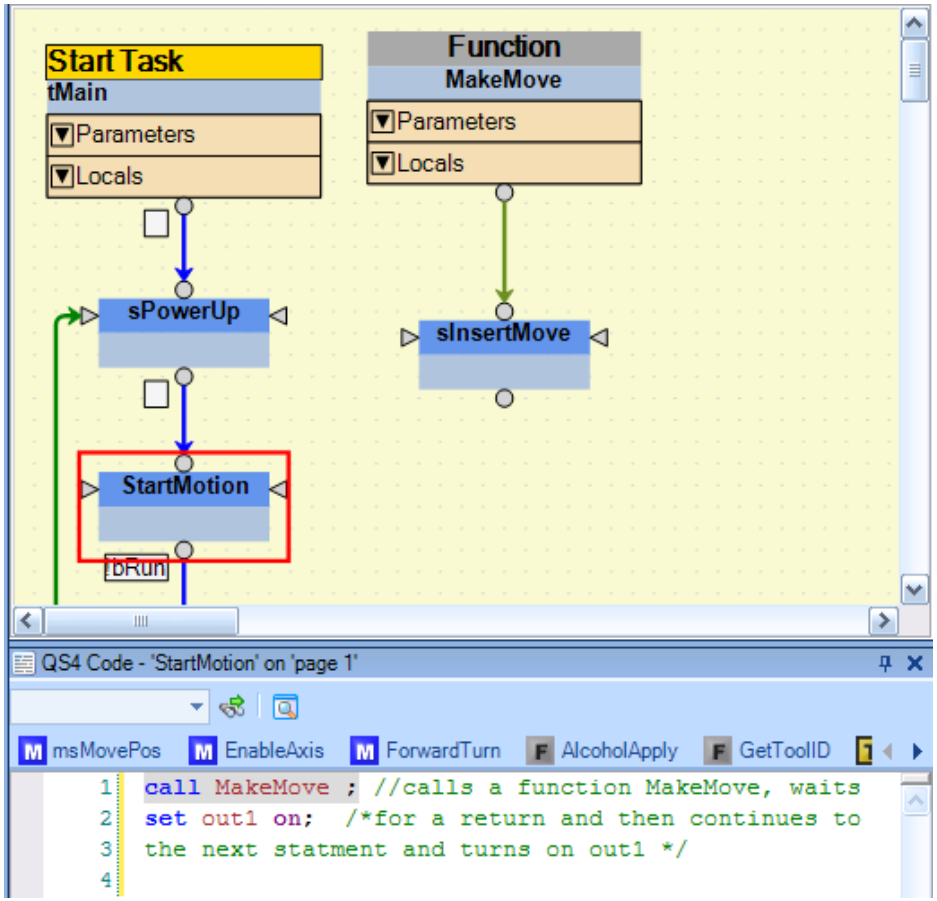
Call

The ***call*** instruction tells the controller to invoke a user-defined function and return control to the next instruction upon completion.

- Completion occurs when the function invokes a return instruction.

call *function*;

We'll take a more in depth look at the call function later.



ctc Control Technology Corp.

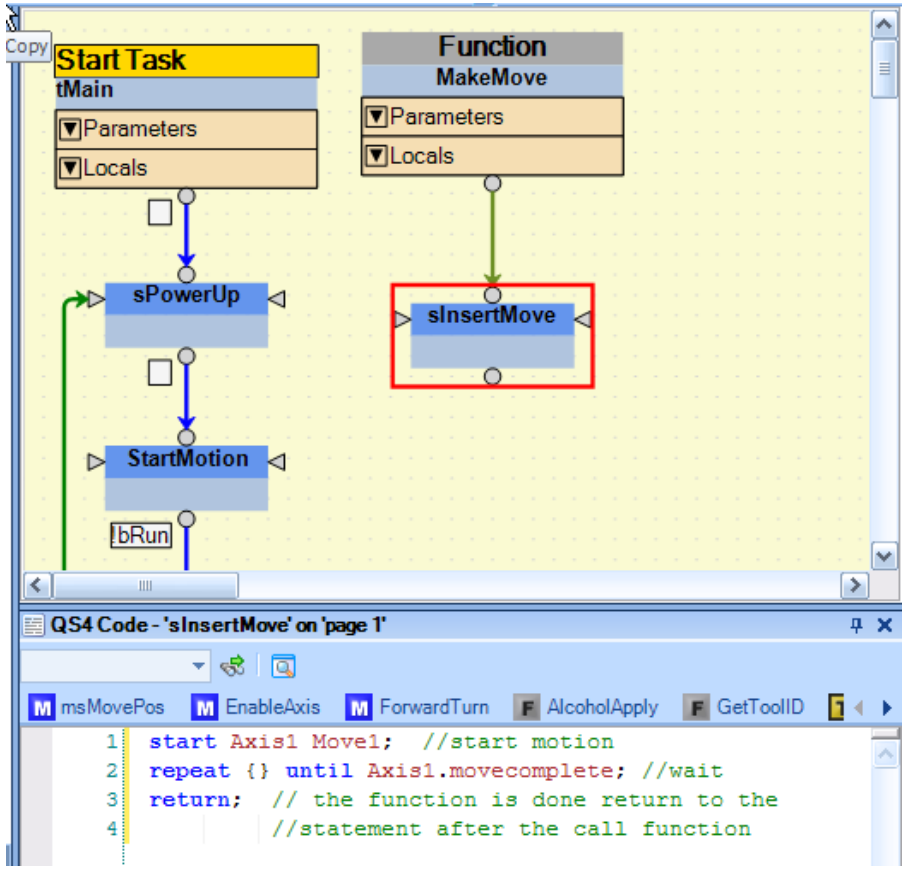
Step Programming

Return

In order to return control (from within the called routine/function), a ***return*** instruction must be executed.

- **return**;

We'll take a more in depth look at the return function later.



ctc Control Technology Corp.

Lab2 – For loops and Functions

The purpose of this Lab is to introduce **Functions** and the use of **For** loops.

Lab2

- We are going to modify Lab1 so that the Start task uses a For loop and Call Statement to run a batch of parts.
- The batch size will be determined by the integer variable BatchSize.

Hint: The existing steps in the Start task will need to be moved to a function.

ctc
Control Technology Corp.

Step Programming

When / Goto

The *when* instruction is *not* equivalent to the *monitor* instruction in QS2.
The *when* instruction *is* equivalent to the *transition condition* used in QS4 steps.

- **when** *expression* **goto** *step*;

```
when count >= 8 then goto ProcessComplete;  /*wait here and then goto ProcessComplete once count >= 8 */
```

ctc Control Technology Corp.

Step Programming

Enable / Disable (Event)

An Event is a task that occurs when a predetermined condition is true. Events must be enabled and often require program-level and disabling and re-enabling.

For example, there may be a specific point in a QuickStep program where an *Event* should not be allowed to occur (initialization, during fault-handling, etc.) This is done with the enable/disable event statements.

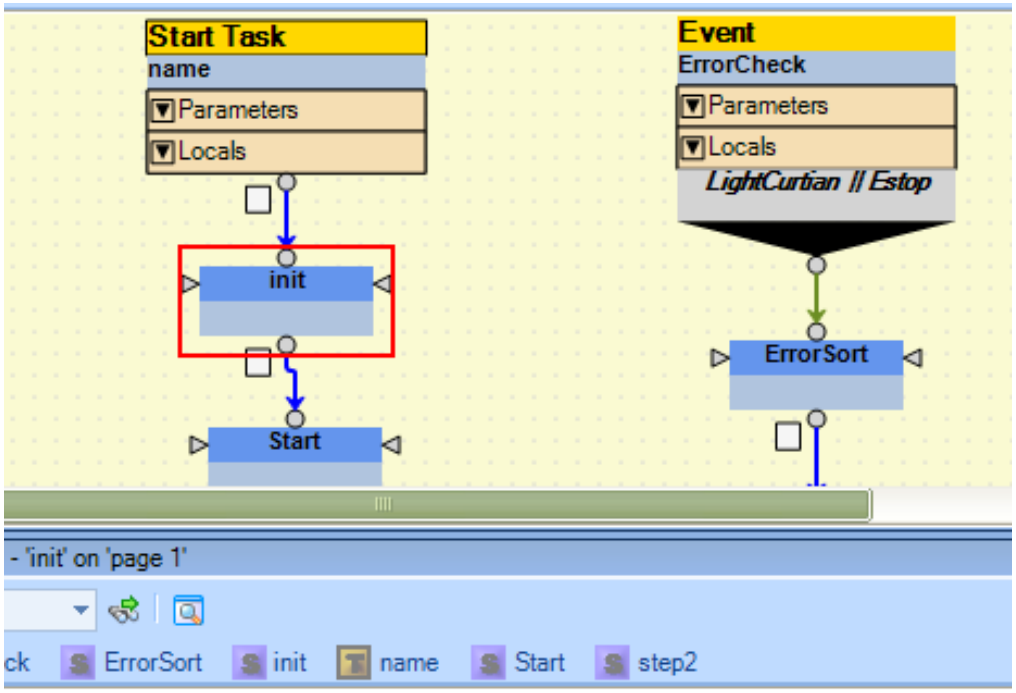
- **enable event** *event*;
- **disable event** *event*;

```
enable event Estop;      // enable the Estop event
```

ctc Control Technology Corp.

Step Programming

Enable / Disable (Event)




```
enable event ErrorCheck;
Speed = 500;
Faults = 0;
```

ctc Control Technology Corp.

Step Programming

Do

The ***do*** instruction starts one or more *child* tasks.

The *parent* task (that executed the ***do*** instruction) will not continue execution until a ***done*** instruction is issued by each child task.

Unlike QS2, do allows you to include other statements within a step.

- **do (** *task task ...* **)**
- **do** *task* **with** expression, expression, …

```
do (Indexer Pick Crimp Vision Eject);   /*Start 5 tasks, wait for them
                                          to finish*/
```

```
do MakeMove with position1, speed1; /*Start the MakeMove task
                                      with the parameters
                                       position1 and speed1*/
```

ctc Control Technology Corp.

Step Programming

Begin

Unlike the ***do*** instruction, the ***begin*** instruction does not wait until the child tasks complete before execution continues. The tasks are started, and control is returned immediately to the *parent* task for continued execution.

When a single task is started with *begin*, the user can store a *task handle* in a *scalar* variable. If multiple tasks are started with *begin*, the user can store the series of *task handles* in a *vector* variable. Various options for using this command are:

1. **begin** *task*;
2. **begin (** *task...* **) task to** *variable* ;
3. **begin (** *task task ...* **)** ;
4. **begin** *task* **task to** *variable* ;
5. **begin** *task* **with** expression, expression, ... ;
6. **begin** *task* **task to** *variable* **with** expression, expression, ... ;

```
begin (Indexer Pick Crimp Vision Eject);   /*Start 5 tasks, and continue to statement4;
statement4;
```

```
begin MakeMove task to MakeMoveTask with position1, speed1;
/*Start the MakeMove task, with the parameters position1 and speed1 and store the
task handle to MakeMoveTask */
```

ctc Control Technology Corp.

Step Programming

Call

The ***call*** instruction tells the controller to invoke a user-defined function and return control to the next instruction upon completion. The ***call*** instruction can pass up to 9 parameters and also return a single *integer* variable.

1. **call** *function*;
2. **call** *function* **store to** *result*;
3. **call** *function* **(***param1*, *param2*, ...**)**;
4. **call** *function* **(***param1*, *param2*, ...**) store to** *result*;

*Calling a function repeatedly, without executing a **return** instruction will result in a stack overflow controller fault.*

```
call MakeMove (position1, speed1);    /*calls a function MakeMove, waits for a
statement20;                            return and then continues*/
```

```
call Calculation (length, width, depth) store to volume;
/*Invoke the Calculation function  using the parameters length, width and
depth and store a result to volume*/
```

ctc Control Technology Corp.

Step Programming

Return

In order to return control (from within the called routine/function), a ***return*** instruction must be executed.

1. **return**;
2. **return** *result*;

The returned *integer* result from the function is stored in the optionally specified variable *result*.

```
return;          /*returns from a function call without the store to option
```

```
return volume;
/*Returns from a function call and returns the integer value volume. Note the name of the store to vs. return value name do not need to match*/
```

ctc Control Technology Corp.

Step Programming

Cancel

The ***cancel*** instruction stops one or more *child* tasks specified in the instruction itself or all running tasks (second form).

The first form of this instruction cancels one or more tasks:

- If the named *variable* is a *vector* variable, then it is assumed to contain a series of *task handles* – each of these tasks will be stopped.
- If the named variable is a *scalar* variable, then it is assume to contain a single *task handle* – the single task will be stopped.

1. **cancel** *variable;*
2. **cancel all tasks**;
3. **cancel other tasks**;

```
if PositionError > 4000 then cancel MakeMoveTask;
```

```
if Estop then cancel other tasks;
```

ctc Control Technology Corp.

Step Programming

Done

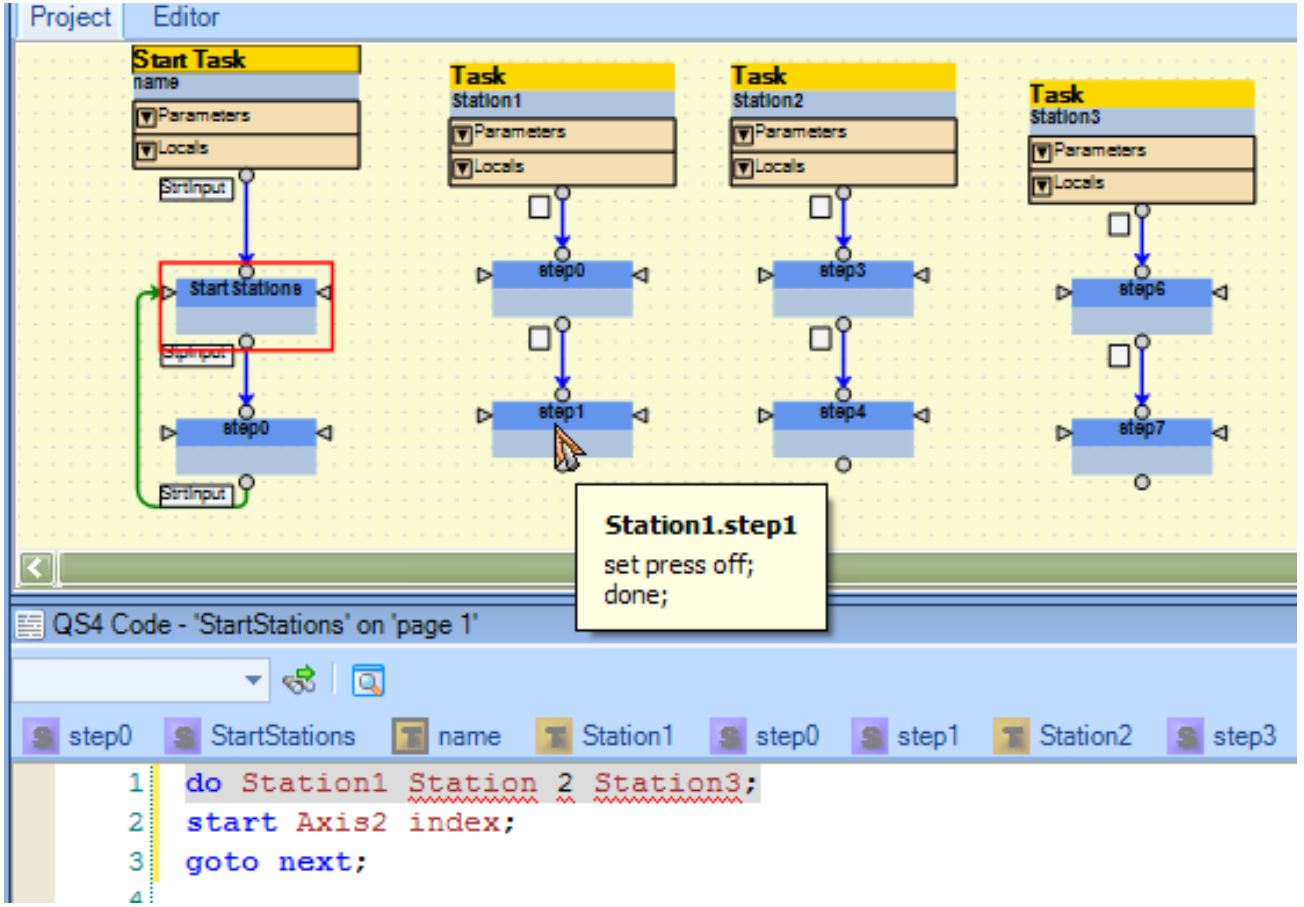
The ***done*** instruction stops *this* task. The task executing the ***done*** instruction. The ***done*** instruction also informs a possibly waiting *parent* task that this *child* task has been completed.

- **done**;

ctc Control Technology Corp.

Step Programming

Done



ctc Control Technology Corp.

Lab3 – Analog I/O and Events

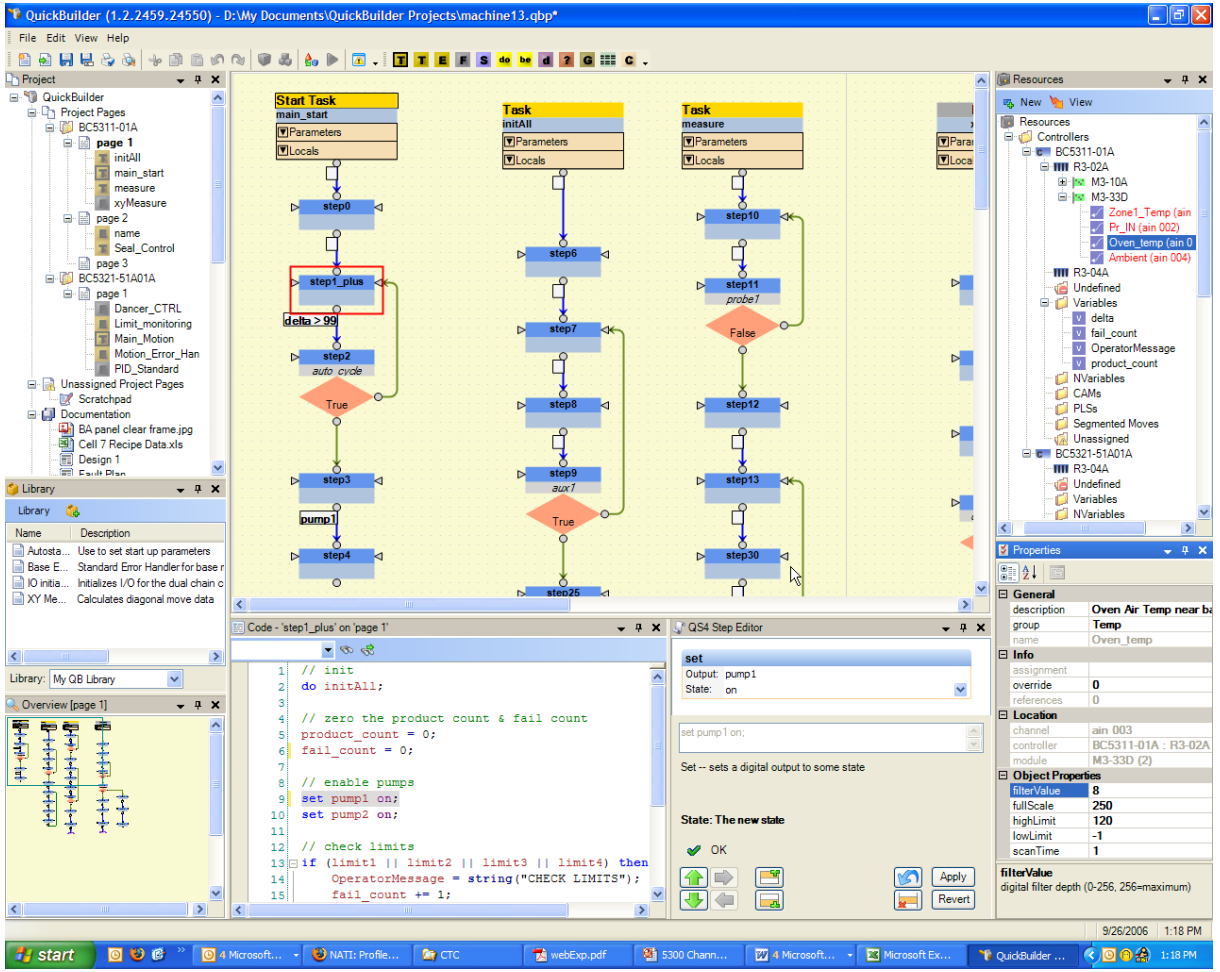
The purpose of this lab is introduce Analog I/O and the use of Events.

Lab3

- We are going to program the controls for a cooling bath. See the Lab handout for more details.

ctc Control Technology Corp.

Additional QuickBuilder Features



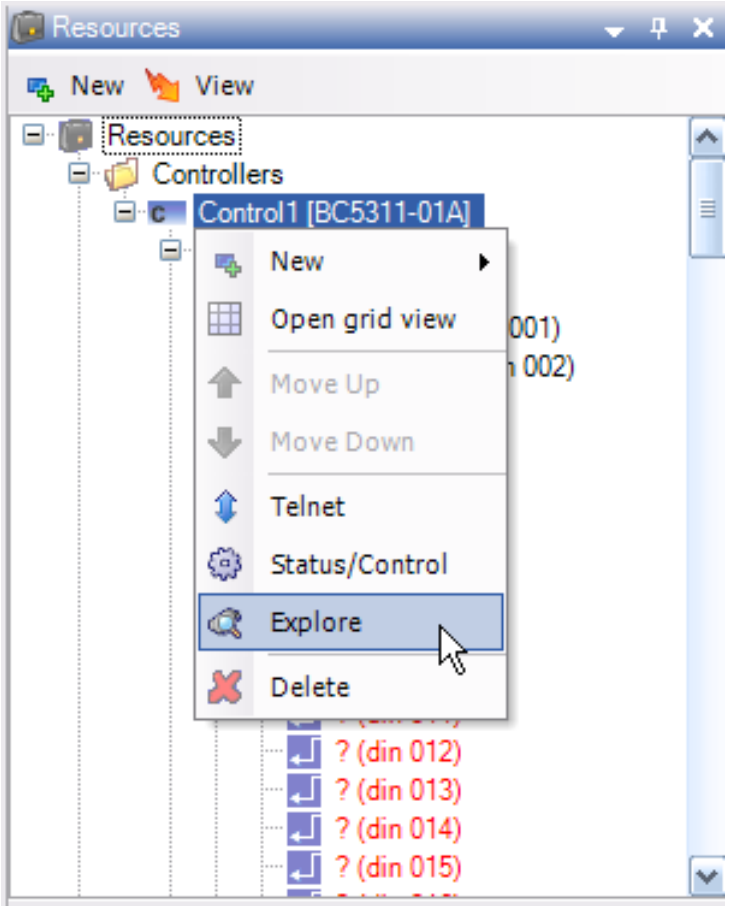
ctc Control Technology Corp.

Additional QuickBuilder Features

FTP Explorer

"File Manager"

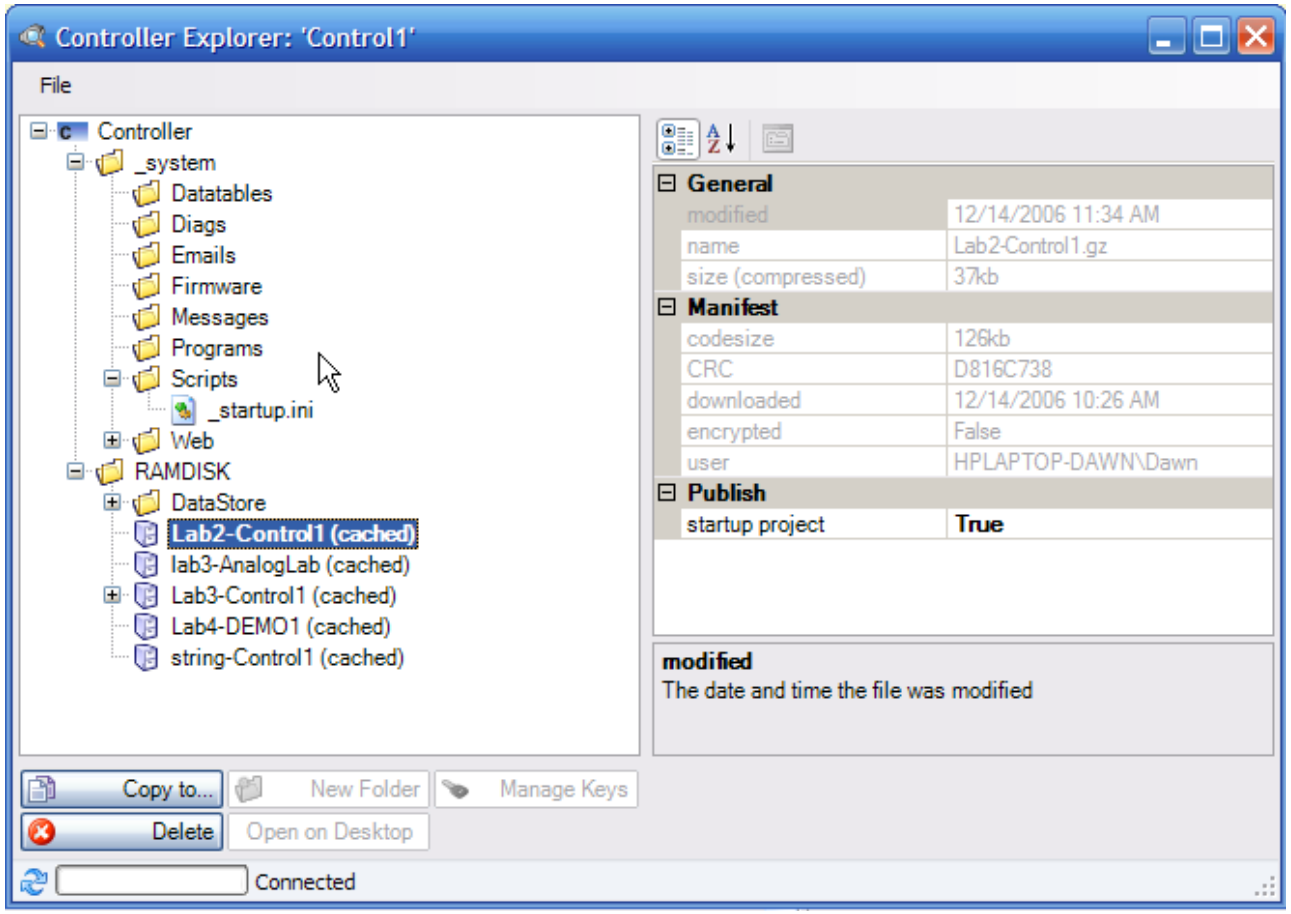
- Organize projects stored on a controller
- Set which project should run when the controller powers-up
- Access via a right click on the Controller and selecting Explore.



ctc Control Technology Corp.

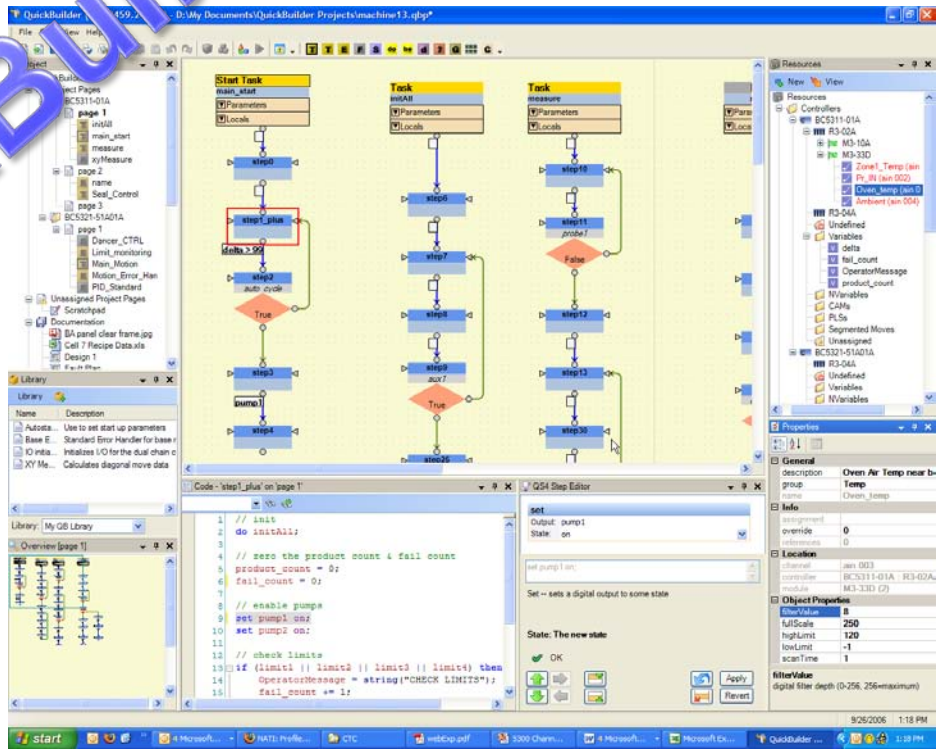
Additional QuickBuilder Features

FTP Explorer
"File Manager"



ctc Control Technology Corp.

QuickBuilder



ctc Control Technology Corp.