

The Progress Electronic Magazine

In this issue:

Publisher's Statement:	2
Coding Article: Using the EQN Package	3
Calling the EQN system with simple algebraic expressions:	3
Adding new functions to the system:	6
Using EQN for business rule formulations:	8
Where to find it:	9
Reference Partners Wanted	10
What is it?.....	10
Who!.....	11
How!.....	12
Publishing Information:	13
Other Progress Publications Available:	13
Products/Services Available From Amduus:	14
Article Submission Information:	14
Order Form for Progress Open Source CD-ROM	15

This document may be freely shared with others without modification. Subscribe for free here: <http://www.amduus.com/online/dev/ezine/EZineHome.html>

Though intended for users of the software tools provided by Progress Software Corporation, this document is NOT a product of Progress Software Corporation.

© Scott Auge and Amduus Information Works, Inc. 2003

Publisher's Statement:

In this issue we explore another piece of software made available to the progress programming community by my company Amduus Information Works, Inc. It is the EQN package – it evaluates algebraic expressions into their computed form.

!!!WANTED!!!

Amduus Information Works, Inc. is looking for consultants to resell access to up-coming ASP web based software. We will need you to find companies who would want use of this software, to configure the software to their needs, and to support them in the use of the software. The software is rented out – no licenses are sold. Each month, you would receive a portion of the revenue, as well be able to bill for training and support – modeled like an insurance agency. Contact sauge@amduus.com for more information.

OK – what does that mean? It means that you can have the user enter $2 + 6$ into a character input widget and after calling EQN it will return 8. As you will see below – even more complex evaluations can be performed. This package is implemented in the Denkh HTML Reporter package – it is the subsystem of code that makes the @calc() macro work. Note that since Denkh HTML Reporter is in use commercially, this code is also being used commercially. It is “production ready.”

Further more – it is explored how to make equations in your application. Instead of placing equations into your code – it may be possible to place equations into your database. Then the code calls out to the database to determine how to calculate something. This can be very powerful for those business rules in your application that depend on equations of known variables.

Alas, I have finished up a contract with the Superior Court of California and am available once again to do work. If any of you have some work that can be done in the United States or via the wire, take a gander at my resume: <http://www.amduus.com/Resumes/ScottAuge.html>. I tend towards web based applications on UNIX/Linux operating systems.

Also, money is getting a wee-bit tight these days in this technology jobs depression. I would like to ask the readers of this publication to send in a donation of maybe \$5.00, \$10.00 or even \$20.00 to aid in the continuation of this publication. Please send these donations to:

*Scott Auge
1818 Briarwood
Flint MI 48507*

Better yet – simply order the CD-ROM available on the last page... Thanks!

Coding Article: Using the EQN Package

By Scott Auge

There are times when one needs to allow the user to enter an expression that should be reduced down to it's computed number. A simple calculator of sorts that a user can enter $2 + 2$ and the code will return 4. Or perhaps a more complex calculator where a user will enter $((3+4)*5)/2$ and the code will return 17.5.

Reach nearly one thousand
programmers and companies.
Your ad could be here!
Advertise in the E-Zine for
\$10.00 per issue!

OK – it is pretty unlikely that you might be writing a calculator in your application – calculators are easily available (though those included on Windows are the more simple type.)

But you could be writing a report writer, such as Denkh HTML Reporter, that uses a macro with an expression that needs to be computed out. Now we are seeing how this could be pretty powerful stuff!

Or your application can include formula's to compute out values for use in the system. For example, a financial analyst determines there could be a formula used in your system that reads:

```
CustomerRisk = CustomerCreditRating * ( 0.10 * LargestPaidOrder)
```

One could easily set up a Formula table where

```
Formula.Name = "CustomerRisk"  
Formula.Formula = "CustomerCreditRating * ( 0.10 * LargestPaidOrder)"
```

When a financial analyst needs a formula changed, so that perhaps the “weight” is 0.12 instead of 0.10, simply change the formula in the record. Sure one could make changes via a parameter record – but this method allows a very dramatic change in formulation to take place without changing the code (provided all the variables are already accounted for in the code.)

First, lets look a more simple invocation of the package using simple algebraic forms.

Calling the EQN system with simple algebraic expressions:

This following program is actually the test program for the EQN package. We use it to fling various algebraic expressions at the code to make sure it parses and evaluates them properly. As you will see, to implement the EQN package is very simple. Once all the program modules are available in the PROPATH, one merely includes one file CalcAlg.i, and calls one function

CalcAlg(). One can also call the Calculate.p procedure if your running into code area space problems in your .r.

```
/*
Copyright (c) 2002 Amduus Information Works, Inc.
scott_auge@yahoo.com sauge@amduus.com

Permission is hereby granted, free of charge, to any person obtaining a
copy of this software and associated documentation files (the "Software"),
to deal in the Software without restriction, including without limitation the
rights to use, copy, modify, merge, publish, distribute, sublicense, and/or
sell copies of the Software, and to permit persons to whom the Software
is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included
in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY
KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE
WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR
PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM,
DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF
CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN
CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER
DEALINGS IN THE SOFTWARE.
*/

{CalcAlg.i}

DEF VAR t AS CHARACTER NO-UNDO.

OUTPUT TO BatCalcAlg.txt.

/* Make sure numbers sent don't blow it up */

ASSIGN T = "42".
MESSAGE "Evaluating " T.
MESSAGE T "=" CalcAlg (T) SKIP.

/* Simple binary operation */

ASSIGN T = "2 + 4".
MESSAGE "Evaluating " T.
MESSAGE T "=" CalcAlg (T) SKIP.

/* Verify Mult and div happen before add and sub */

ASSIGN T = "2 + 4 * 3".
MESSAGE "Evaluating " T.
MESSAGE T "=" CalcAlg (T) SKIP.

/* Negative numbers work */

ASSIGN T = "2 + -4".
MESSAGE "Evaluating " T.
MESSAGE T "=" CalcAlg (T) SKIP.

/* Grouping operators work in expected order */
```

```
ASSIGN T = " ( 2 + 4 ) * 3".
MESSAGE "Evaluating " T.
MESSAGE T "=" CalcAlg (T) SKIP.

/* More complicated group */

ASSIGN T = " ( 2 + 4 ) * 3 * ( 7 + 3 ) ".
MESSAGE "Evaluating " T.
MESSAGE T "=" CalcAlg (T) SKIP.

/* Nested group orders work */

ASSIGN T = " ( 2 + 4 * ( 3 + 3 ) ) * 3 * ( 7 + 3 ) ".
MESSAGE "Evaluating " T.
MESSAGE T "=" CalcAlg (T) SKIP.

/* Take a shot at functions */

ASSIGN T = "Power[ ( 3 + 4 ) ,2 ]".
MESSAGE "Evaluating " T.
MESSAGE T "=" CalcAlg (T) SKIP.

ASSIGN T = "2 * Power[ ( 3 + 4 ) ,2 ]".
MESSAGE "Evaluating " T.
MESSAGE T "=" CalcAlg (T) SKIP.

ASSIGN T = "Power[ ( 3 + 4 ) ,2 ] * 2".
MESSAGE "Evaluating " T.
MESSAGE T "=" CalcAlg (T) SKIP.

ASSIGN T = "Power[ 7,2 ] * 3".
MESSAGE "Evaluating " T.
MESSAGE T "=" CalcAlg (T) SKIP.

ASSIGN T = "Power[ 7,3 ]".
MESSAGE "Evaluating " T.
MESSAGE T "=" CalcAlg (T) SKIP.

ASSIGN T = "Power[ Power[ 2,2 ] ,4 ]".
MESSAGE "Evaluating " T.
MESSAGE T "=" CalcAlg (T) SKIP.

/* Test new expression cleaner uper */

ASSIGN T = "(3+4)*5)/2".
MESSAGE "Evaluating " T.
MESSAGE T "=" CalcAlg (T) SKIP.

ASSIGN T = "(2*(2+4))".
MESSAGE "Evaluating " T.
MESSAGE T "=" CalcAlg (T) SKIP.

OUTPUT CLOSE.
```

Here are the results of the above program.

```
Evaluating 42
42 = 42
Evaluating 2 + 4
```

```

2 + 4 = 6
Evaluating 2 + 4 * 3
2 + 4 * 3 = 14
Evaluating 2 + -4
2 + -4 = -2
Evaluating (2 + 4) * 3
(2 + 4) * 3 = 18
Evaluating (2 + 4) * 3 * (7 + 3)
(2 + 4) * 3 * (7 + 3) = 180
Evaluating (2 + 4 * (3 + 3)) * 3 * (7 + 3)
(2 + 4 * (3 + 3)) * 3 * (7 + 3) = 780
Evaluating Power[(3 + 4), 2]
Power[(3 + 4), 2] = 49
Evaluating 2 * Power[(3 + 4), 2]
2 * Power[(3 + 4), 2] = 98
Evaluating Power[(3 + 4), 2] * 2
Power[(3 + 4), 2] * 2 = 98
Evaluating Power[7, 2] * 3
Power[7, 2] * 3 = 147
Evaluating Power[7, 3]
Power[7, 3] = 343
Evaluating Power[Power[2, 2], 4]
Power[Power[2, 2], 4] = 256
Evaluating ((3+4)*5)/2
((3+4)*5)/2 = 17.5
Evaluating (2*(2+4))
(2*(2+4)) = 12

```

Adding new functions to the system:

As you can see in the above test code, there is a function available called `Power[]`. Functions need their arguments enclosed by a `[` and a `]`. The magic for creating functions for use in expressions is contained in `AlgFunc.i`.

```

/*
Copyright (c) 2002 Amduus Information Works, Inc.
scott_auge@yahoo.com sauge@amduus.com

Permission is hereby granted, free of charge, to any person obtaining a
copy of this software and associated documentation files (the "Software"),
to deal in the Software without restriction, including without limitation the
rights to use, copy, modify, merge, publish, distribute, sublicense, and/or
sell copies of the Software, and to permit persons to whom the Software
is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included
in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY
KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE
WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR
PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM,
DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF
CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN

```

```

CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER
DEALINGS IN THE SOFTWARE.
*/

/*****
/* Define your function here */
*****/

/* Sample function with two arguments */

FUNCTION Power RETURNS CHARACTER (INPUT cArgs AS CHARACTER):

    DEF VAR cArg1 AS CHARACTER NO-UNDO.
    DEF VAR cArg2 AS CHARACTER NO-UNDO.

    DEF VAR dResult AS DECIMAL NO-UNDO.

    /* Args come in a single string delimited with , */

    ASSIGN cArg1 = ENTRY (1, cArgs)
           cArg2 = ENTRY (2, cArgs).

    /* Do your computation */

    ASSIGN dResult = EXP(DECIMAL(cArg1), DECIMAL(cArg2)).

    /* Return a string! */

    RETURN STRING (dResult).

END. /* FUNCTION Power */

/*****
/* Add your function to be evaluated here. */
*****/

FUNCTION FuncEval RETURNS CHARACTER (INPUT cExpression AS CHARACTER ):

    DEF VAR cSubExpression      AS CHARACTER NO-UNDO.
    DEF VAR iSubExpressionIndex AS INTEGER NO-UNDO.
    DEF VAR cReplaceExpression  AS CHARACTER NO-UNDO.
    DEF VAR cArgExpression      AS CHARACTER NO-UNDO.
    DEF VAR cEvalExpression     AS CHARACTER NO-UNDO.

    /* We always evaluate functions from the right to the left */
    /* to make sure they are evaluated according to algebraic */
    /* rules. We repeat to make sure we get em all! */
    /* To add your own function: */
    /* 1) Create a function as above */
    /* 2) Determine the name of the function in the expres- */
    /* sion */
    /* 3) Add AlgFuncTemplate.i First argument is the ex- */
    /* pression name, the second is the function name in */
    /* in your code. */

    /***** Begin Example Template *****/

    {AlgFuncTemplate.i Power Power}

    /***** End Example Template *****/

```

```
RETURN cExpression.  
END.
```

The code is fairly self-documenting.

One would need to make a function accepting a character input and returning a character input. The input string will be the function's arguments. They will appear in the same order as presented in the equation. Since the evaluator is working on strings, be prepared to do some variable type conversions.

Once your function is ready, you need to install a call to it in the table of "function name" to "function routine." It is simple to do this. In the FuncEval() function, include the AlgFuncTemplate.i file with two arguments as presented in the above code.

Your done! You can now use your function in the expressions.

Using EQN for business rule formulations:

As teased in the introduction to this article, one may want to set up a more complex system that uses formula's to compute out values.

Just as a financial analyst might use a spreadsheet to design a model, one can include those spreadsheet formula's in an application based on that model¹. Doing so, the program should be much more tightly coupled with the ideas expressed in the spreadsheet.

Once a formula has been set up, you can call out as shown below:

```
/* Compute out the risk of this customer not paying for an order.      */  
  
/* Find the formula associated with customer order risk of non-payment */  
/* This formula should be based on data mining to figure out a spread */  
/* comparing customer attributes to non-paid orders and paid orders.  */  
  
FIND Formula NO-LOCK  
WHERE Formula.Name = "CustomerRisk".  
  
ASSIGN cFormula = Formula.Formula.
```

¹ Of course an application will need to be coded for specific variables in the model. One may need to adapt the application to include variables found in the spreadsheet model, or limit the spreadsheet model to those variables available in the application.

```
/* Determine the variables used in the formula.  These names will be used */
/* later on to figure out what their values are.  They are returned in a */
/* comma delimited list.                                                    */

ASSIGN cVars = FindVariables(cFormula).

/* Variables used through-out the model are computable here.  This receives */
/* a list of variables, and knows how to populate them with numbers.  The */
/* population might be dependent on record look-ups, other formula's,      */
/* initialization files, etc.  The routine will return the values in a      */
/* comma delimited list with respect in order to the variable names given. */

RUN CalcVars.p (INPUT cVars, INPUT cCustomerNumber, OUTPUT cValues).

/* Now we substitute the variable names with the variable values.  Notice */
/* how we have spacing around the variable names.  All variables must be */
/* wrapped in spaces.  This prevents accidental replacement of variables */
/* who's name happens to include as a substring the name of another var- */
/* ible.  Be sure to relate this rule to formula entry people.            */

DO i = 1 TO NUM-ENTRIES(cVars):
  cFormula = REPLACE (cFormula, " " + ENTRY(i, cVars) + " ", " " + ENTRY(i,
cValues) + " ").
END. /* DO i = 1 TO NUM-ENTRIES(cVars) */

/* All the variables should have values - Call out to the EQN package */
/* to compute out the value.                                             */

ASSIGN dValue = DECIMAL(CalcAlg(cFormula)).
```

Where to find it:

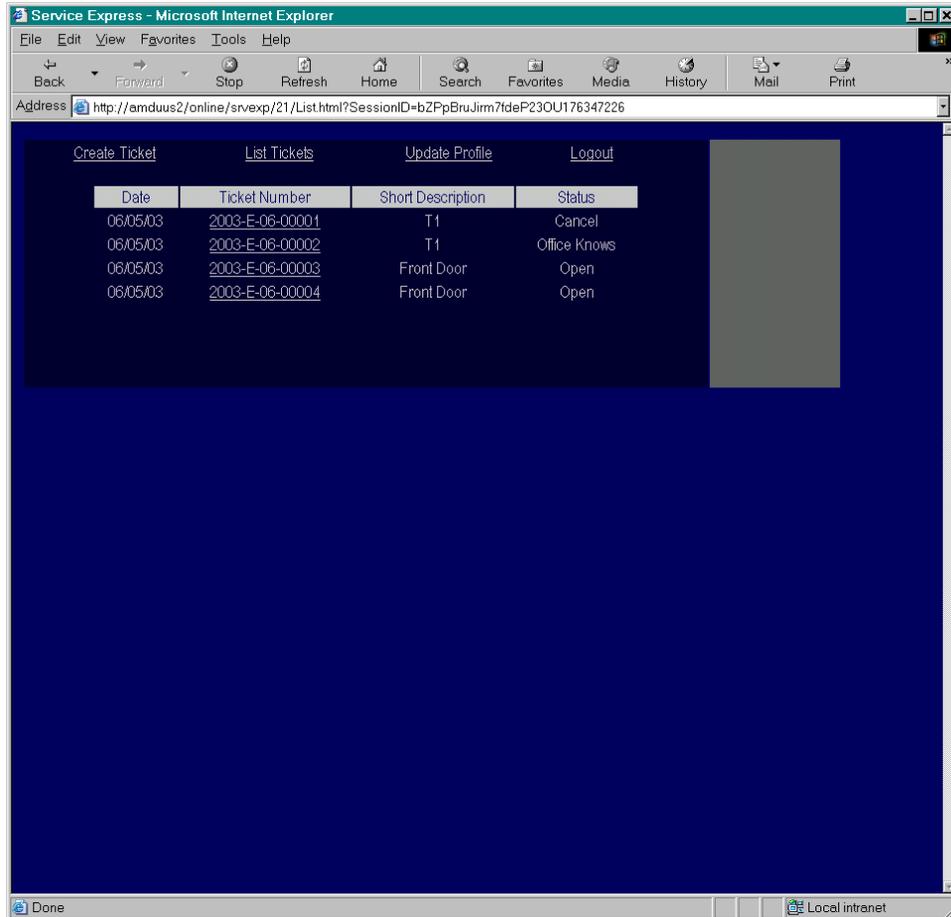
<http://ns1.amduus.com/OpenSrc/SrcLib/EQN/>

About the author: Scott Auge is the founder of Amduus Information Works, Inc. He has been programming in the Progress environment since 1994. His works have included E-Business initiatives and focuses on web applications on UNIX platforms.

sauge@amduus.com

Reference Partners Wanted

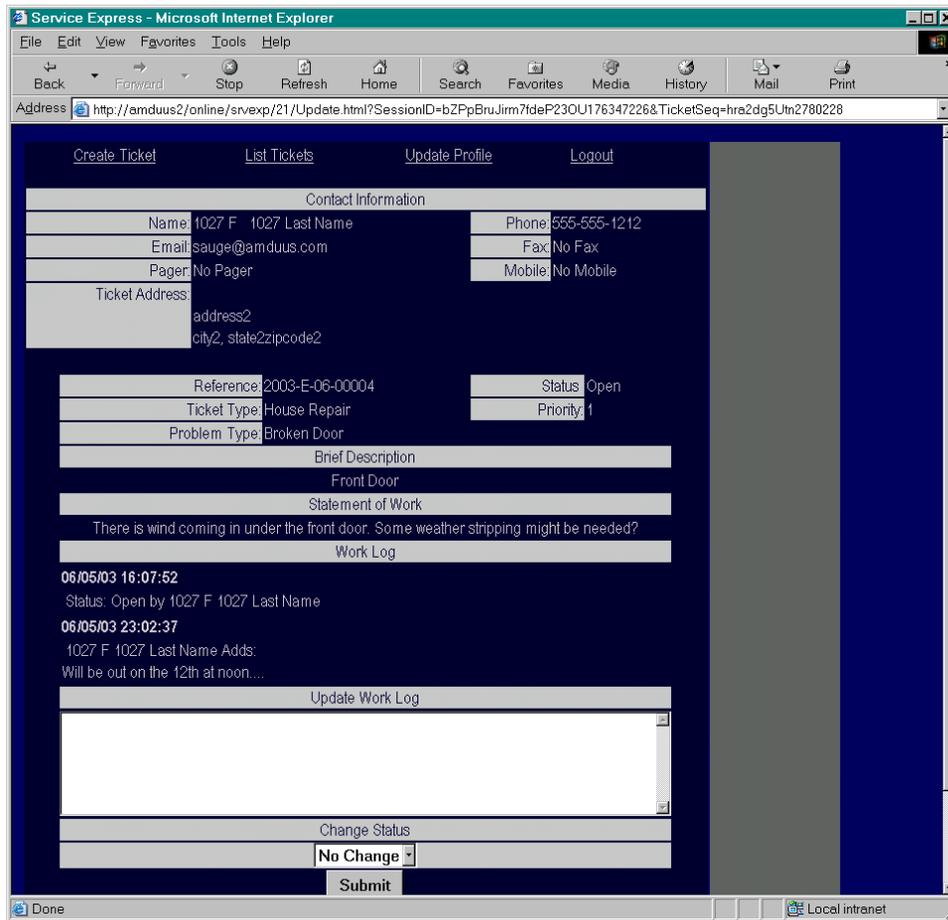
Amduus Information Works, Inc. finally has a working version of the Service Express available!



A listing of tickets belonging to a customer

What is it?

What is it? It is a work order/issue tracking system. It lets your customers (whether they be your co-workers, or the company's customers) create work orders (tickets) to solve problems. The tickets can be categorized by types and problems; And a work flow can be developed to track the status of the ticket. All of this is configurable to match your industry or user base.



A customer's ticket information and work log

Who!

We would like to make it available to five companies/organizations that might be willing to offer suggestions for improvement and are willing to be referenced as user's of the software.

The company/organization can be governmental, non-profit, private, or a corporation.

Amduus Information Works, Inc. also provides documentation services! Scott Auge notes, "One of the things I have noticed throughout my contracting career is that companies with developed software always seem to be missing or weak on user documentation, administration documentation, and programmer documentation." Amduus can help you with this!

How!

You can reach the public portion of a demo for the application here:

<https://www.amduus.com/cgi-bin/se0001pub/21/index.html>

You would be acting as a customer of a company called “Demo” with relations to the organization running the web site. Your company name would be Demo, and to prove that you are indeed an employee/representative of that company, you would know that the authorization code is Demo.

Let me point out that you could be an internal customer, such as HR approaching IT about setting up a new employee; Or, you could be an external customer approaching the organization with a request for a service to be performed.

This could be used by a manufacturing company for repairs/over-hauls of their equipment. An apartment complex for handling property issues. Any company/organization that wishes to interact with it’s customers in a manner that needs to be defined and tracked for a process of completion.

Be sure to see the accompanying Power Point presentations for more information.

Please contact Scott Auge at sauge@amduus.com if you have additional questions!

Publishing Information:

Scott Auge publishes this document. I can be reached at sauge@amduus.com.

Amduus Information Works, Inc. assists in the publication of this document by providing an internet connection and web site for redistribution:

Amduus Information Works, Inc.
1818 Briarwood
Flint, MI 48507
<http://www.amduus.com>

Other Progress Publications Available:

This document focuses on the programming of Progress applications. If you wish to read more business oriented articles about Progress, be sure to see the Profile's magazine put out by Progress software <http://www.progress.com/profiles/>

There are other documents/links available at <http://www.peg.com> .

There is a web ring of sites associated with Progress programming and consultants available at <http://i.webring.com/hub?ring=prodev&id=38&hub> .

Products/Services Available From Amduus:

Amduus Information Works, Inc. is a Progress reseller and ASPen partner. We primarily develop UNIX/Linux based applications with web interfaces for manufacturing, service, and law enforcement communities.

We also perform integration of Progress applications to non-Progress applications through such languages and tools as MQ Series, C, and C++.

Amduus provides support for the following applications: Blue Diamond, Denkh, Denkh HTML Reporter, Red Arrow Portal (CMS), Survey Express and other software.

Amduus is looking for consultants who might want to promote the use of our tools at user groups and companies they might work in. Send some information to sauge@amduus.com to let me know you are out there!

Article Submission Information:

Please submit your article in Microsoft Word format or as text. Please include a little bit about yourself for the About the Author paragraph.

Looking for technical articles, *marketing Progress* articles, articles about books relevant to programming/software industry, white papers, etc.

Send your articles to sauge@amduus.com! Thanks!

Order Form for Progress Open Source CD-ROM

COUPON 001A

This is an offer for the CD-ROM at lower than list savings!
This is a great way to support the E-Zine too!

Mail this form to:
Amduus Information Works, Inc.
1818 Briarwood
Flint, MI 48507



Please send _____ copies of the Open Source CD-ROM at \$25.00 per disk to:

Name _____

Company _____

Address _____

City _____

State _____ Country _____

Zip _____

Please make your checks/money orders out to: [Amduus Information Works, Inc.](#) Cash works too!
This offer only valid in the United States of America and those countries with postal agreements with the United States Post Office.

The CD-ROM includes (all source code included):

- Blue Diamond/IRIS – Webspeed alternatives
- Survey Express – easily create text templates of surveys and then have the program generate the web pages automatically
- Service Express – Web based Help Desk.
- The Progress E-Zines, books on learning to program in Webspeed (PDF/Word/HTML)
- Denkh HTML Reporter – web based report writer
- CMS – a web content management system
- DB Email – Use pop3 to download emails into a Progress database
- Neural Networks – experiments in spam recognition and text message classification
- Denkh – create PDF file reports for Webspeed/UNIX CHUI!

- [More!](#)