PROCEDURES FOR PROGRAM PUBLICATION THROUGH EXATRON AND ESFOA

This document is to serve as a guide for those programmers who wish to submit original software to be considered for publication by Exatron on a royalty payment basis. In addition, guidance will be presented to those Exatron Stringy Floppy Owners Association (ESFOA) members who wish to donate software to their local ESFOA library or direct to the National ESFOA library.

This guide will be divided into three parts as follows:

Part I	Procedures for submitting software to Exatron for publication on a royalty basis.
Part II	Procedures for submitting software to an ESFOA library on a donation basis.
Part III ...	Suggestions on programming techniques and formats.

INTRODUCTION

ESF owners face two software problems. The first is program selection and/or adaption; the second is program creation.

There are many excellent programs available for the TRS-80 and when transferred to ESF wafer, they will meet the varied needs of many ESF owners. However, adapting some of these programs to the ESF format so that they may be transferred from a cassette based program to an ESF based program requires, in some cases, substantial programming skill. Exatron invites those ESF owners who can do this to share their solutions with other owners whether or not they seek payment for their contributions. As rapidly as such information is submitted and reviewed, it will be distributed to workshop chairmen and other ESF owners. All owners will benifit by a maximum interchange of information in this manner.

In encouraging this way of solving problems of program transfer to wafers, Exatron does not condone misuse of copyrighted programs. However, wafer storage is superior to audio cassette storage and ESF owners are entitled to use wafers for all programs they have legitimately acquired.

The second problem is program creation. Many programs remain to be written. When an ESF owner has developed one he thinks will be of use to others, we encourage submittal of such a program to Exatron or ESFOA. The following pages will outline the procedures for these submittals. We look forward to receiving your work for review and hope that a mutual financial rewarding relationship will develope.

PART I

PROCEDURES FOR SUBMITTING SOFTWARE TO EXATRON

Part I will cover the procedures that a programmer must follow when submitting software to be considered for publication by Exatron under a royalty system of payment. Part I will be divided into three sections which will cover the following information:

Section A ... Program Submittal
Section B ... Documentation
Section C ... Program Acceptance

SECTION A

PROGRAM SUBMITTAL

The following steps should be observed in submitting a program to Exatron:

1. Submit a signed Software Review Agreement. This assures us that the program is your creation, to sell or give away as you see fit. WE WILL NOT EVALUATE A PROGRAM WITHOUT THIS ASSURANCE. This Software Agreement is attached to the end of this document. Make as many copies of this form as you desire.

2. Submit your program on wafer with a label indicating title and file number. If the program contains more than one file, then be sure the label is identified accordingly. If the program is accepted, the wafer will be replaced, otherwise, it will be returned. If you submit a patch to a commercial program that is not sold by Exatron, submit as a separate package, the original, unmodified program. We cannot evaluate patches without the original program and in most cases we will not have such programs available.

3. Submit the program documentation as requested in Section B.

4. Indicate the names and qualifications of one or more persons who have received and used the program.

5. The standard royalty arrangement at Exatron is a 50-50 division of the sales price less $2.00 for the wafer. Alternatively, if desired, the programmer's royalty can be specified as a set amount of money for each copy sold. This arrangement will be modified (see Software Agreement) when the program is to be sold as part of a reduced price promotional package or is to be resold through an Exatron sales representative. In the latter case, the sales representative would be buying programs in quanity from Exatron and would of course be eligible for a discount price.

SECTION B

DOCUMENTATION

The nature, length, and purpose of your program will determine how much information you should send in. Most of what follows applies to full programs. Program patches or modifications should require less detail. Your documentation should be two separate pieces of work. One for the review staff who will evaluate your software and another different set of documentation that will go along with your program for the end user. The required contents for these two sets of documentation are as follows:

FOR THE REVIEW STAFF.

This document is your sales pitch to us. It should be neat and tend to create in the minds of the reviewer a favorable impression of what your program might be like when he gets around to running it. Who knows, if your documentation is attractive enough the reviewer just might put your program a little ahead of some of the others to be reviewed. This documentation should be typed (or printer output in upper/lower case) and be double spaced with wide margins. Be sure your typewriter or printer has a new ribbon. The documentation will have to be reproduced for the review staff and a light printout will not reproduce well.

Include a covering letter with your program documentation covering the following points:

1. What your program does.

2. Why it should interest others and how it will benefit them.

3. In what respects is your program better than others of the same type. (I wonder how many checkbook, inventory, mail list, shoot the airplane, and guess the word programs there are available).

4. A thumbnail description of your program that you would like to see in our Software Catalog.

Unless the program is completely self documenting, the program documentation itself should cover the following detailed information:

1. A description of what the program intends to accomplish and a listing of any special memory or peripheral requirements the user must have in order to successfully run the program.

2. If the program requires the loading of any special utilities before your program may be run, be sure and point this out right at the beginning. For example ... Data I/O, printer driver, upper/lower case driver, special data wafer, etc.

3. A detailed, step by step set of instructions on just how to run the program. This is the most important part of the documentation and can make or break your entire programming efforts. The program code itself may be the greatest thing ever written, but if you cannot lead the user by the hand in showing him how to execute it properly every step of the way, then the program is useless.

This part of the documentation should take the user from the moment of power up right through to the conclusion of the program run. Leave nothing out. As a general rule in writing your documentation never assume anything. You will not insult someone's intelligence by describing in detail how some steps should be performed in the operation of your program. The knowledgeable person realizes that you must document for all levels of expertise and the novice will be grateful that you did. You can be sure that good documentation associated with your name on a program will assure interest and credibility in your future works.

4. Give examples throughout your documentation on what might be expected to be seen on the screen. Try to anticipate questions and give clear explanations throughout. Have others read your documentation and improve upon it through the benifits of their misunderstandings and questions.

5. Include your address on the documentation so that users may know how to reach you in case of difficulty. Some authors also include a phone number where they can be reached and this is very thoughtful. This of course is up to you, but an address is essential. Keep in mind that if your documentation is clearly and well written, there won't be much time spent on your part answering much more than fan mail.

6. If applicable (such as a business type program) the documentation should contain instructions on how to modify or customize your program for varied application purposes. In addition you might want to include a program listing with comments on what each section does and a separate list of variables and their meanings.

7. If your program has provisions for outputing to a printer, your documentation should indicate which type printer it supports. Point out where in your program the printer routine is located so that a user can modify it, if neccessary, so that the program will output to his printer which may be of a different type than the program was designed to support. You might want to include the type of printer support in your thumbnail description of the program. Programs that output to a printer seem to generate the most questions from users because of the proliferation of types of printers available and their different column widths and control codes.

FOR THE END USER.

Unless the program is completely self documenting, the end user documentation should include the following:

1. The program documentation from above including any agreed upon suggestions for changes that may have been put forward by the reviewers. The documentation should be single spaced and in upper/lower case new ribbon printout or typewritten.

2. This documentation should be presented in its final form in the exact format you would have liked to have received the documentation yourself if you were the buyer of the program. All that should be necessary on our end after acceptance, is the reproduction of your program and its documentation.

3. This end user documentation need not be submitted until the program has been accepted and all neccessary changes made.

4. If the program is completely self documenting, then a short covering letter to go along with the program to explain any loading instructions might be all that is neccessary.

SECTION C

PROGRAM ACCEPTANCE

After your program and supporting documentation have been received, Exatron will begin evaluation. The evaluation procedure will normally be accomplished in five phases as described below.

1. Your submission will be logged in and perused for proper submittal procedures. If the material appears to have been submitted in accordance with this document, the program and documentation will be passed on to a programmer for the next phase.

2. At this point a programmer will check the operation of your program against the documentation. He is not evaluating the program for accuracy in its subject matter or computed results; this will be done as a part of phase 3.

During this phase of the evaluation the programmer is merely trying to get an idea of what the program is supposed to do and asertain if the documentation is being presented in a clearly written style. He will be checking for obvious mistakes in the documentation and for gross poor programming techniques in the operation of the program. There won't be any nitpicking here. Just a common sense run through of the operation to make sure the program isn't bombing out right and left and the screens aren't full of sloppy, poorly planned displays. Part III will offer some suggestions to help you through this phase of the evaluation.

If the programmer now feels your work has merit and is of professional calibre, he will have copies made of your program and documentation (maximum of 10) and will pass your work onto others who are knowledgeable in the subject of your program.

3. Reviewers who are knowledgeable in the subject of your program will now subject your complete work to a thorough workout and will return a completed reviewers report to the programmer.

4. A consensus of the programmer's and reviewer's reports will be returned to the author along with a letter stating whether or not Exatron desires to publish your program. If the reviewer's reports are favorable and Exatron feels your program has commercial potential, then you will be asked to make any suggested changes and return the program and documentation in its final end user format for publication.

5. Upon acceptance of your final version of the program, Exatron will forward to the author a completed Disclosure Agreement indicating acceptance of the program along with the mutually agreed financial arrangement for royalty payments.

Exatron receives exclusive rights to distribute the program on ESF wafer, but you retain the right to sell it on any other media. If at some later time you choose to have the source code published in a paper or magazine then your publishing agreement with Exatron is automatically cancelled.

Exatron will distribute the program on ESF wafer, with suitable documentation and packaging. The primary means of sale will be by listings in the Exatron Software Catalog.

Exatron will account for sales and pay royalties due at the end of each calendar quarter. The agreement reached may be terminated without cause or penalty by either party after 120 days written notice and in the case of the source code being published in a paper or magazine, the agreement will become terminated forthwith.

PART II

PROCEDURES FOR SOFTWARE SUBMITTAL TO AN ESFOA LIBRARY

When an ESF owner has developed a program he feels will be of use to others but does not wish to go through the rigors of having it published commercially, we encourage submittal of such a program to an ESFOA library on a donation basis. Procedures for submittal are given below. Documentation instructions follow. A Disclosure Agreement is enclosed separately at the end of this document.

PROGRAM SUBMITTAL

All donated programs must be submitted to your local ESFOA workshop librarian. If you do not know where your nearest workshop is located, then contact Exatron for the information. Donated programs will only be accepted by the National Library at Exatron when it is determined that there is no workshop within your geographical area.

1. Submit a signed Program Donor Agreement. This assures us that the program is your creation, to sell or give away as you see fit, or is a program in the Public Domain. (See NOTE at the conclusion of this Part II). ESFOA WILL NOT ACCEPT ANY PROGRAMS WITHOUT THIS ASSURANCE. (Make copies of the enclosed Program Donor Agreement if you desire).

2. Submit your program on wafer and include loading instructions. If the program is accepted the wafer will be replaced, otherwise it will be returned.

3. Submit the documentation requested below.

DOCUMENTATION

The nature, length, and purpose of your program will determine how much information you should send in. It should be typed (or printer output) with a good ribbon and include the following:

1. What the program does and how to run it. Spare no details. Lead the user through every step of the program run.

2. Memory size, language, and equipment requirements.

3. Include a thumbnail description of your program that you would like to see in the Library Software Catalog.

4. If you desire to have your donation considered for acceptance to @LOAD, then your documentation should be a little more extensive and include tutorial information on how different sections of your program code executes and such other information you can include to assist the user of your program in learning from your programming techniques.

For programs being donated to the library it is suggested that they be of the self-documenting type or require very little support documentation. This will help keep costs down in reproducing the software for distribution to others. This suggestion of course does not neccessarily hold for those programs donated in the hope that they may be selected for @LOAD.

Upon receipt of the above items, the local ESFOA library review staff will review your program and, if accepted, a gratuity will be sent to you in appreciation for your donation. If, in the course of the evaluation, your program and documentation is deemed to be of a quality suitable for @LOAD, a recommendation will be made to the local librarian to forward the program to the National Library for consideration to appear in a future issue of @LOAD. If your program is accepted for @LOAD, you will be issued a certificate worth up to a $500.00 credit towards the purchase of any product marketed by Exatron.

*** NOTE ***

Programs keyed in from magazine listings may be submitted as original software and be eligible for consideration for @LOAD if the following requirements are met:

1. The program must be substantially modified by you and a description of your modifications should be included in your accompanying documentation. Describe how your modifications resulted in an improvement over the original program.

2. Include in the documentation the name, address, subscription rates,* and the publication date of the magazine in which the original program appeared. Following that information, give the name and address (if known) of the original author along with a statement such as: "Modified by (your name, address, and date)".

3. When the program is RUN, the first thing to appear on the screen should be the credits as described above. You may, however, delete the magazine address and subscription rates.

4. Include no documentation that is a copy from the magazine article. The documentation must be in a completely original form.

* Subscription rates can usually be found on the Table of Contents page of the magazine and it is a desired procedure to mention this information along with any reference to a magazine.

PART III

SUGGESTED PROGRAMMING TECHNIQUES AND FORMATS

Unfortunately, many worthy programs must be returned to their authors for rework simply because their programming techniques and screen formatting were not of professional calibre. You might be thinking at this point; "Well I'm not a professional programmer. I'm a hobo by trade and I've written this neat little program that will figure the best times and places to hop a freight. How then can I ever expect my program to be accepted for publication?"

The point to be made clear here is that you don't have to be a professional programmer by trade to write a program in an acceptable professional like style. It's really very simple and just takes a little further study on your part and some common sense. The extra effort required in preparing your work to be published as a commercial piece of software, is necessitated by the fact that once you ask for and receive compensation for your work, you remove the protective cloak of an amateur and join the ranks of professionals. Your work will now be judged and evaluated against higher standards and it is the purpose of Part III to offer some suggestions that might help in making this transition.

Before getting into some suggestions for you to think about, let's carry the above case of the hobo a little further to point out what all too often happens when an amateur or novice decides to submit a program that he wants to sell.

In a typical case, a covering letter will usually accompany the program extolling its virtues. This is great and the reviewers certainly want to be enlightened on the subject of your program. In the case of our hobo friend he probably has told us how about a year ago he saw a need for a program like this and decided to write one himself because there wasn't anything on the software market that accomplished what he wanted. He goes on to say how he's been using his program for many months now and he's never miscalculated a freight yet. His buddy has used it and swears by it. In fact, it's mainly at the urgings of his friend that he has decided to offer it for sale.

So far so good. It appears that the author has something worthwhile and the reviewer can hardly wait to type RUN. (Provided of course that the accompanying documentation is understandable, but that's a whole other story). The results of this first evaluation is disaster. How can it be disaster you say when the program has been used for many months and there is his friend's testimonial to proove how good it is.

The answer lies in the fact that he sent in a program that he has coded for his own use only. No thought was given to the fact that possibly a non programmer or novice might be using his program. There were no efforts made to user proof it. That is, program traps in the code to catch improper key entries, or error trapping techniques to prevent loss of data. (Extremely important on data base programs). He has probably forgotten about the times he verbally clarified some points to his friend and failed to upgrade the program accordingly.

The INKEY$ function was never used, only the simpler INPUT function was used with its inherent abilty to louse up screen displays with strange messages like EXTRA IGNORED and REDO. Not even a CLS statement was used in the first line of the program so that after entering RUN, all the previous garbage remained on the screen while the computer went into its apparent hang up act while the program went through its initialization routine.

There were long delays in the execution of many phases of the program. Even selecting a menu choice took an inordinate amount of time for the program to execute that choice. Inspection of the code revealed the fact that no variables were initialized before arrays were dimensioned. This of course slowed down program execution tremendously. Also it was noticed that GOTO and GOSUB statements were being referenced to REMARK statements rather than lines following the REMARK statements. Here again is another cause of slow execution.

There didn't seem to be any thought given to making the screen displays pleasing to look at. PRINT @ statements were hardly ever used. Everything seemed to be crunched up in the upper left side of the screen and there were spelling errors and gross mistakes in grammer.

All in all the first impression was that the programmer did not have much pride in his work. No apparent effort was made to upgrade the program from an amateurish home brew effort to a work of class that would be readily accepted by any user as a first rate program that was worth the money he would pay for it. Now the program did do what it was supposed to and obviously the author knew his subject, but for it to be in a class (professional) where he could expect to ask and receive money for it, a litle something more needs to be done.

It's not enough to simply take a barebones program you have written and used successfully yourself and offer it for sale. A little extra effort must be made to make it look professional as well as making sure it executes properly with efficient code.

Usually a person programming for himself won't take the time to put in a lot of user proofing and cosmetics since after all, he knows what he's supposed to do all the time and who cares what the displays look like so long as they get the message across. But when it comes time that you would like to sell it to others, think about some of the following suggestions for upgrading and polishing. I'm sure it will go a long way towards getting you past that first commercial software review.

USER PROOF IT

This is the most important step in preparing your program for commercial sale. User proofing means to code your program in such a way that unexpected answers to program questions or accidental incorrect data entries won't cause the program to crash or go into some far out operation that has nothing to do with the desired function at hand.

Keep in mind that your program may be used by people who are not programmers and may even be new to the whole world of computer usage. This concept is especially important to keep in mind if your software is of a business type and may be used by inexperienced office personnel.

Use of the INKEY$ function and error traping routines are two of the most powerfull tools you have at your disposal in accomplishing this step and should be used whenever feasible.

When you use INKEY$ inplace of the INPUT function you gain almost complete control over what will be accepted as correct input by your code associated with INKEY$. As an example, if your program is looking for numeric entries and some other key is accidently pressed, your code with INKEY$ can prevent that incorrect entry from even appearing on the screen. An INPUT statement, on the other hand, would end up giving error messages on the screen and possibly even causing the screen display to scroll which could result in the loosing of some important information off the top of the screen. Not only that, but your beautiful screen layout could be completely disrupted. Press the CLEAR key by mistake when in an INPUT situation and see what happens. Remember you must press the ENTER key to activate an INPUT statement and the CLEAR key is right next to it.

This brings out another useful feature of INKEY$. Whenever a single key entry is all that is required to activate certain parts of your program such as a (Y/N) answer or selecting from a menu that has no more than nine choices, your INKEY$ code can have the entry itself activate that portion of your program. This will obviate the necessity of having to always press ENTER every time. This can make your program progress smoothly and quickly through its various routines.

Of course there are some cases where the INPUT statement is more feasible or possibly the author wants you to take that extra step of pressing ENTER before a critical choice is made, but generally you can code the INKEY$ function to take care of most of these situations and it does make for a more closer control on what is being entered. The choice here will mainly be determined by you and what you want the program to respond to, but give serious thought to using INKEY$ whenever you can feasibly do it. It will require more programming thought on your part but will usually result in a more professionally executed program.

Error trapping routines can go along way in preventing program crashes and possible loss of data. Use them wisely and they will help keep your program intact and easily recoverable from certain errors that might be made. Also your documentation can be of tremendous help here in guiding the inexperienced user in recovering from errors.

COSMETICS OR "ADDING A TOUCH OF CLASS"

Although of less importance than the preceeding USER PROOFING suggestions, adding a touch of class to your programming efforts will go a long way in establishing that look of professionalism in your program displays and execution. Although the following suggestions for polishing up your program won't have any affect on the outcome of the program RUN, they will add that extra touch that generally takes them out of the look of an amateur effort and into the look of a professional piece of work.

1. Always begin your program with a CLS command. If your program is going to take a little time to get going due to some initialization routines, then print a message telling the user what's happening. Make your program friendly and helpful throughout.

2. Make your message and instruction displays pleasing to read and look at. Center your displays and make different portions look in balance with the whole screen. Pay particular attention to spelling and gross grammatical errors. Always have others proof read your program as well as your documentation. Use the 32 character mode whenever your text will allow. Nothing looks so amateurish and lazy than to see a short message squeezed up in the upper left corner of the screen. You have 1024 bytes of screen memory at your disposal. Take advantage of all that free memory to add a touch of class to your displays.

3. If your program uses arrays, always initialize your variables (in decending order of usage as much as possible) before you dimension the arrays. This will cause your program to execute much faster as the computer will not have to take the time to move up all the arrays everytime you introduce a new variable. You can set numerics to 0 and strings to null if you want to. The point is to mention them before the arrays are set so that they will have their places already reserved in the variables lookup table.

4. Using REMARK statements in your program is an excellent idea but be carefull where they are located in relation to GOTO and GOSUB statements. A REMARK statement line number should never be executed as this needlessly takes up program execution time. Try to have your initialization routines and display text programming residing near the end of your program and your most often used routines residing towards the beginning of your program. This will also help in speeding up program execution.

5. One last important point. If you modify your program in any way, be sure and rerun the program completely from the beginning. So many times a last minute change is made that you know is only going to affect one portion of the program and that is the only part that is retested. Unfortunately this alteration can create a problem elsewhere in your program that just might never have crossed your mind. A complete rerun will flush out any newly hatched bugs.

It is sincerely hoped that these few suggestions will help in speeding your program submissions through the review process and result in high quality programs for us all. This is the goal of the Software Division at Exatron.

SOFTWARE REVIEW

Before a program can be considered ready for release to the buying public, Exatron requires the software and its documentation to meet certain minimum standards. A volunteer review staff, consisting of people with varied professional backgrounds, is used by our software division in assisting them in the important task of selecting software to be marketed by Exatron.

The bottom line and primary objective is to do everything possible to make sure that the buyer of any software from Exatron is going to get his money's worth. That means programs and documentation written in a professional manner and sold at a reasonable price. To meet this objective, certain guidelines are provided the review staff in order that they may objectively carry out their evaluations of submitted programs. These guidelines are as follows:

THE PROGRAM

1. Be user friendly.
2. Execute in an effecient and logical manner.
3. Well error trapped and user proofed. Does not bomb on unexpected or incorrect inputs.
4. Pleasing and well thought out text/graphic displays.
5. Correct grammar and no spelling errors in text displays.
6. Absolute minimum or zero use of INPUT statements. INKEY$ statements are preferred.
7. Execution and screen displays conform with documentation instructions and examples.
8. Program subject must conform to Exatron's software marketing plans.

THE DOCUMENTATION

1. The final version must be camera ready copy in upper/lower case printout.
2. No gross grammatical or spelling errors.
3. Must be written with the thought in mind that the user is totally unfamiliar with the program and will need detailed guidance on getting it up and running and using it in the manner envisioned by the author.
4. To support the concept in item #3 above, all documentation should include sections with the following headings:

   (A) INTRODUCTION. Here is where you give the user an overview of what the program is about and what he should and should not expect from the program.
   (B) SYSTEM REQUIREMENTS. A listing of the hardware, memory, and any special software that is required.
   (C) INSTRUCTIONS FOR MAKING A BACKUP COPY OF THE PROGRAM.
   (D) PROGRAM CUSTOMIZING. If applicable, instructions on

making any program changes that may be required due to differences in applications or equipment installations.

(E) USING THE PROGRAM. Here you will lead the first time user through the execution of the program from power up to its conclusion. Liberal use of examples can be of tremendous help here.

If the program is of such complexity and length to warrant it, then a Table of Contents and an Index might be necessary. Also a separate quick reference section might be a welcome addition so that the experienced user of the program won't have to wade through detailed instruction text for a quick memory refresh.

Finally, all documentation should contain the author's address so that the user will know where to seek help should that need ever arise.

We expect our reviewers to be candid in their remarks and to let the chips fall where they may. Only then, after resolving the review comments with the author (throwing out suggestions there and accepting others here) can we expect to fulfill our objective.

Many times authors new to the review process by "strangers", tend to take negative comments about their program as personal attacks upon themselves and to their programming abilities. I can assure you that this is never the case. The reviewers are doing their job to help get the best product possible for the ultimate buyer. Their job is to criticize and their criticisms may be both positive and negative. They are judging the product and not you personally. So as hard as it may seem at first, put away your personal feelings and realize that we are all on the same team working together on a common objective. Remember, if your submission was not well thought of in the beginning, it would never have been sent on to the review staff for their thoughts on possible improvements.

When you receive the review results, look them over objectively, and don't be afraid to admit to yourself that maybe some of the suggestions are good ones and should be implemented. You'll probably want to kick yourself at times for not having thought of some of these things yourself. Of course some suggestions may be completely off base from the objectives of your program and you should feel just as free to reject those comments and suggestions. Clearly stating the program objectives in your INTRODUCTION will help keep the reviewers on track in this regard.

I hope these few paragraphs will help clear up some of the misconceptions concerning the "why" of a review and also to let you know that even reviewers get reviewed when they submit a program and boy you ought to see the fireworks then. But you know what? .... We sure get a final good piece of software that the author and Exatron can be proud of. And that's what it's all about.

SOFTWARE REVIEW and ACCEPTANCE AGREEMENT
(COMMERCIAL SOFTWARE ONLY)

1. ...., hereinafter referred to as programmer, in consideration of the mutual covenants herein contained, agrees as follows to provide for evaluation the following program, and by signing this agreement warrants that he has good title to this program and that said program is entirely his own original work. Programmer further certifies by his affixed signature that no other firm, corporation, or person has any interest, right, or claim to said program or any part thereof.

Program Title: .... Suggested Retail Selling Price:.........

2. Exatron agrees to evaluate said program for consideration as being acceptable for publication and if so deemed acceptable, will publish, distribute and sell such program on Exatron Stringy Floppy wafer media.

3. Exatron agrees to pay royalty of 50% of the net receipts from the sale of said program. Net receipts shall be gross receipts minus $2.00 per wafer. Gross receipts shall be calculated on the retail selling price as listed in the Exatron Software Catalog. Exatron shall have the right to reduce this retail selling price by no more than 40% whenever bulk sales are made to Exatron sales representatives for resale or during certain promotional package sales deemed advisable by Exatron to assist in increasing the software sales volume. Promotional package sales will not be offered for more than a 60 day period nor will any specific program be offered in a promotional package at any less than 6 month intervals. Gross and net receipts shall be calculated, and said royalties shall be paid to the programmer at least once every 3 months or at such lesser intervals as Exatron, in its sole discretion, shall determine.

4. The programmer agrees to provide Exatron with as much comment and documentation as is practical including instructions required by users of the program and a copy of the program on ESF wafer in Radio Shack Level II or Radio Shack Level II plus Microsoft Level III BASIC or in Assembly Language or Machine Code readable on Radio Shack TRS-80* Mod.1 Level II computers. The programmer will also assist in the preparation of the program for distribution including making changes to the program. The programmer agrees to provide full disclosure of any changes or updates made in the program and further agrees that said changes or updates may be made in the program copies being distributed by Exatron.

5. Exatron reserves the right to make up to ten copies of the program and supporting documentation for evaluation purposes.

6. The programmer will retain the right to sell or distribute by any means, other than ESF wafer, the program contracted for herein.

7. This agrrement may be terminated without cause or penality by either party after 120 days written notice or in the case wherein the program source code has been published in any paper or magazine, this agreement will be considered terminated forthwith.

....
Programmer Date

Address ....

City, State, Zip ....

Telephone ( ) ....

Program accepted by:

....
BILL BURNHAM, Manager Software Division, Exatron Corp. Date

* TRS-80 is a trademark of Radio Shack, Tandy Corp.

PROGRAM DONOR AGREEMENT
(PUBLIC DOMAIN SOFTWARE ONLY)

1. ...., hereinafter referred to as programmer, in consideration of the mutual covenants herein contained, agrees as follows to provide for evaluation the following program, and by signing this agreement warrants that he has good title to this program and that said program is entirely his own original work or is a program in the Public Domain. Programmer further certifies by his affixed signature that no other firm, corporation, or person has any interest, right, or claim to said program or any part thereof.

Program Title: ....

Indicate which applies:
Original Work ....
Public Domain in its original form ....
Modified Public Domain ....

2. ESFOA agrees to evaluate said program for consideration as being acceptable for inclusion in the ESFOA Library. It is further agreed that if ESFOA determines that said program is acceptable for @LOAD that ESFOA will make availabe to the programmer a certificate worth up to a $500.00 credit towards the purchase of any product marketed by Exatron Corp. It is further agreed that in consideration for having donated a program, and its eventual acceptance, ESFOA will make available to the programmer/donor a certificate worth $9.95 in credit towards the purchase of any product marketed by Exatron Corp. ESFOA shall have sole discretion in determining whether a program is acceptable for donation to its library.

3.The programmer agrees to provide ESFOA with as much comment and documentation as is practical including instructions required by users of the program and a copy of the program on wafer in Radio Shack Level II or Radio Shack Level II plus Microsoft Level III BASIC or in Assembly Language or Machine Code readable on Radio Shack TRS-80* Model 1 Level II computers.

....
Programmer Date

Address ....

City, State, Zip ....

Telephone ( ) ....

Program accepted for Library by:

....
Librarian ESFOA Chapter No.

City, State, Zip ....

Copy to: Programmer/Donor
National ESFOA Librarian

* TRS-80 is a trademark of Radio Shack, Tandy Corp.