University of Michigan
Space Physics Research Laboratory

Software Development Notes	CAGE No.	0TK63
	Drawing No.	055-3934
	Project	TIDI
	Contract No.	NASW-5-5049
	Page	1 of 5

REVISION RECORD

Rev	Description	Date	Approval
-	Initial Release		

APPROVAL RECORD

Function	Name	Signature	Date
Originator	M. Burek		
Flight Software	S. Musko		
Software Manager	D. Gell		
Program Manager	C. Edmonson		
Systems Engineer			
R&QA	John Eder		

Contents

Figures

Tables

1. References

(1) Production Control Requirements TIDI document 055-TBD

2. Introduction

The purpose of this document is to describe the procedures that are used during TIDI software development to enhance the maintainability and accessibility of the code generated during the project. Procedures for makefiles, the MKS archival system, and release are covered.

3. Procedures

3.1 *makefile procedures*

The release section of a makefile is normally added to the makefile after the compilation and linking sections of the file. This section copies executables from the development area to the released software area. It is desirable to include statements in the makefile that allow the "release" section of the makefile to operate from only from the project directory, not “sandbox” (code development) areas. In order to do this a variable is created (PROJECT_SOURCE below) which holds the path to the project directory in full. A MKS .IF statement in the makefile that checks that the current working directory is equal to the project source directory. If it is the copy is allowed, if no match an error message is generated. Below is an example of the syntax required:

```
PROJECT_SOURCE = /tidi/tidi_software/dbQuery
EXE = path to the release directory

release:
.IF $(MAKEDIR) == $(PROJECT_SOURCE)
    cp dbQuery $(EXE).
    chgrp tidi $(EXE)dbQuery
    chmod 2755 $(EXE)dbQuery
    chmod g+s $(EXE)dbQuery
.ELSE
    @echo "ERROR: Can not release this program from this directory."
    @echo "       It must be released from the main project directory"
$(PROJECT_SOURCE)
.END
```

Note that in MKS makefiles the .IF, .ELSE and .END statements *must* start in the first column.

3.2 *Adding information to source and executable files*

The MKS system allows release note information to be added to source files. The information may be embedded in a comment or a string. Currently we are embedding two types of information.

3.2.1 Log information in comment strings.

The $Log$ command is put in a string command:

```
c, c++        /*
                *$Log$
                */

FORTRAN        c   $Log$

PERL           # $Log$

TICL           ;  $Log$
```

The $Log$ keyword must be capitalized. This keyword is replaced by the version information during the check in of the source file. The information that is inserted is the comment, date of check in, and the person doing the check in. As the program is released multiple times, a list of releases grows to show the information from all releases.

3.2.2 Embed information about the release.

Currently we are embedding the Compile time and date, the revision number for each file and the revision level for the project in a form that can be recovered using the UNIX "what" command. The technique to accomplish this is to start each string that is to be displayed with a "key" sequence, currently "@(#)" Examples are given below for c, FORTRAN and PERL.

c or c++

```
char ident[] = "@(#)sourceFileName.c - File $Revision$\n";
char ident2[] = "@(#)sourceFileName.c - $ProjectRevision$\n";
#define dt "@(#)sourceFileName.c - Compile Date/Time: " __DATE__ " " __TIME__
"\n"
char ident3[] = dt;
```

FORTRAN

```
character*256   IDENT/"@(#) sourceFileName.f - File $Revision$>"
character*256   IDENT/"@(#) sourceFileName.f - $ProjectRevision$>"
```

PERL

```
#  $ident1 = @(#)sourceFileName.pl - File $Revision$\n";
#  $ident2 = @(#)sourceFileName.pl - $ProjectRevision$\n";
```

3.3 "Usual" overall release procedure

For the purposes of this discussion the assumption is made that the reader is familiar with the MKS tools and has checked out, modified and tested code that now needs to be released. The steps I follow are below.

1. Open MKS and the sandbox project file for the file(s) to be released in the sandbox area.
2. Check in the modified file(s) to the sandbox archive.
3. Open the corresponding master project in /tidi/tidi_software/"your_project"
4. Highlight the modified file(s), and click the resynchronize button . The "delta" icon should go away, and there should be a message in the MKS window advising that the working version is now the latest version.
5. In the main project directory type "make" to build the master project, or use the MKS make button to recompile and link the project, for code that must be compiled.
6. When the program compiles correctly, test it for the changes that were made
7. Type "make release" to copy the executable program to the released software directories.
8. Retest the executable again from a directory different from the executable's directory.
   Exception: The EPETs which must be executed in the executable directory