{Introduction}
{ACI}

ACI stands for \hA\hdvanced \hC\hommunication \hI\hnterface, and it allows robot users to communicate to a robot controller using an IBM-PC or other host computer. The ACI protocol checks for communication errors, and allows general access to the robot memory for data transfer.

\mFile Utilities\m
\mSetup\m
\mMemory\m
\mTerminal Emulation\m
{}

{File Utilities}
{1}
\hFile Utilities\h

All common file transfer utilities and other file utilities are contained in this menu selection. Transfer to and from the robot, as well as file edit and directory selection items are available here. Shelling out to the DOS system is available also.

\mSend\m	\mReceive\m	\mTokenize\m
\mUntokenize\m	\mRenumber\m	\mEdit\m
\mPCP Load\m	\mDirectory\m	\mOS Shell\m
\mQuit\m		

{}

{File Send}
{Send}
{1.1}
\hFile Send\h

Send a file to the robot memory. Files can be program, variable, location, or hex type data files, and are selected from a further menu.

\mProgram Send\m
\mVariable Send\m
\mLocation Send\m
\mJob Send\m
\mHEX Send\m
{}

```
{Program Send}
{1.1.1}
              \hProgram File Send\h

All .ROB files that are available in the current
directory are displayed, and can be selected for
transfer to the robot.
     \mQuick Search\m
     \mFile Send\m
     \mFile Types\m
{}

{Variable Send}
{1.1.2}
              \hVariable File Send\h

All .VAR files that are available in the current
directory are displayed, and can be selected for
transfer to the robot.
     \mQuick Search\m
     \mFile Send\m
     \mFile Types\m
{}


{Location Send}
{1.1.3}
              \hLocation File Send\h

All .LOC files that are available in the current
directory are displayed, and can be selected for
transfer to the robot.
     \mQuick Search\m
     \mFile Send\m
     \mFile Types\m
{}


{Job Send}
{1.1.4}
               \hJob File Send\h

A single .JOB file in the current directory can
be selected for transfer to the robot.
     \mQuick Search\m
     \mFile Send\m
     \mFile Types\m
{}

{HEX Send}
{HEX File Send}
{1.1.5}
```

\hHex File Send\h

A single .HEX file in the current directory can be selected for transfer to the robot.
\mQuick Search\m
\mFile Send\m
\mFile Types\m
{}

{Receive}
{File Receive}
{1.2}
\hFile Receive\h

Data files can be created on the disk using the robot memory as the source of information.
\mProgram Receive\m
\mVariable Receive\m
\mLocation Receive\m
\mJob Receive\m
\mHEX Receive\m
{}

{Program Receive}
{1.2.1}
\hProgram File Receive\h

All .ROB files that are available in the robot directory are displayed, and each can be selected for transfer to the IBM-PC.
\mFile Receive\m
\mFile Types\m
{}

{Variable Receive}
{1.2.2}
\hVariable File Receive\h

A file name is entered as the target for all robot variable data. The entire variable table contents is transferred to the IBM-PC.
\mFile Receive\m
\mFile Types\m
{}

{Location Receive}
{1.2.3}
\hLocation File Receive\h

A file name is entered as the target for all robot location data. The entire location table contents is transferred to the IBM-PC.

```
     \mFile Receive\m
     \mFile Types\m
{}


{Job Receive}
{1.2.4}
              \hJob File Receive\h

A job file is created by selecting the potential
files for storage from the robot, and then all of
the variables and locations are stored in
separate files with the same name as the
job file.
     \mFile Receive\m
     \mFile Types\m
{}


{HEX Receive}
{1.2.5}
              \hHex File Receive\h

A single .HEX file in the current directory can
be selected for transfer to the robot. The user
is prompted for a starting address for the
transfer as well as the length of the transfer
in bytes. The file is saved using the standard
Intel HEX file format.
     \mFile Receive\m
     \mFile Types\m
{}


{1.2.5.1}
             \hHex File Receive Offset\h
{}

{1.2.5.2}
           \hHex File Receive Size\h
{}


{TOKENIZE}
{1.3}
               \hFile Tokenize\h

A program file or a group of program files is
selected for tokenizing.  Tokenizing replaces the
command in each RAPL-II program line with a
number which can be decoded more quickly than its
alphabetical counterpart. The result is a RAPL-II
```

program that executes more quickly (10x faster on average). When the \mDEFAULT\m ON selection in the SETUP EDIT window is ON, then the original file name is used for the tokenized result, and the same name with a BAK extension is used for the original file. If the DEFAULTS ON is not selected, then the system prompts for a new destination name for each file in turn.

\mFile Utilities\m

\mFile Types\m

{}

{1.4.1}
{1.3.1}

\hFile Selection\h

Hit the SPACE key or the ENTER key to select one or more files. Selecting "\mALL BUT\m", reverses the meaning of the marks - marked files are not included and are not processed.

{}

{UNTOKENIZE}
{1.4}

\hFile Untokenize\h

For debugging purposes, it is often useful to untokenize a previously tokenized file. This action will restore the command name in each file. Like the \mTOKENIZE\m selection, the file selection is identical.

\mFile Utilities\m

\mFile Types\m

{}

{RENUMBER}
{Renumber}
{1.5}

\hFile Renumber\h

When generating RAPL-II programs, it is often useful to renumber a file so that new lines can be inserted with new line numbers, or it is common to renumber a file once all editing is completed in order to give the file an evenly numbered appearance. With \mDEFAULT\m ON, the current setting of starting line number and line increment will be used, as well as the original file name. Selecting DEFAULTS OFF will

force the user to enter these values each time,
or at least to confirm their values.

See the \mStart Line\m and \mLine Increment\m
cells in the environment configuration block

See also \mFile Types\m
         \mFile Utilities\m
{}

{1.5.1}
           \hProgram Renumber File Selection\h

Hit the SPACE key or the ENTER key to select any
file. Selecting "\mALL BUT\m", and then any file will
exclude those files from being processed.
     \mRenumber\m
     \mFile Types\m
{}

{Text Editor}
{Edit}
{1.6}
             \hFile Edit (Alt-E)\h

A text file editor of the user's choice will be
executed from this selection. The file editor is
specified in the SETUP EDIT window. The only
limitation to the editor that is selected is that
it must fit into the DOS memory available once
RobComm-II is loaded. The file name being edited
is maintained after each selection, so it is very
convenient to move in and out of the editor. The
file editor can be invoked using the ALT-E hot
key selection.

See also \mEdit File\m

{}

{PCP Load}
{1.7}
                \hPCP File Load\h

Process Control Programs are a special type of
program that is loaded into the robot memory.
A PCP file is a HEX file type that is created
from a parent .EXE file residing in the current
data directory.

Using technical information from CRS Plus, a programmer can use any high level language to program the robot controller to do a variety of custom applications, or even to replace RAPL-II itself. Advantages in the execution speed of compiled programs will permit certain types of functions to be performed more easily this way.

\mFile Types\m
{}

{Directory}
{1.8}
\hFile Directory\h

The default data directory can be changed by selecting it here. Also, changing the directory when in DOS will have the same result.
{}

{OS Shell}
{1.9}
\hOperating System Shell (Alt-D)\h

The DOS COMMAND.COM program will be invoked, and the user can proceed to execute any DOS command or utility that can reside in the available memory. DOS can be invoked using the ALT-D hot key selection. Certain DOS functions can be programmed in the SETUP FUNCTIONS menu, and they can be executed using the CNTRL-F1 to CNTRL-F10 hot key selections.
{}

{Quit}
{1.10.1}
{1.10}
\hQuit RobComm-II (Alt-X)\h

Release all memory used by RobComm-II, and return to the parent process, which is usually DOS itself.

The ALT-X hot key selection will automatically terminate RobComm-II.
{}

{SETUP}
{2}

\hRobComm-II Setup\h

All operating characteristics of RobComm-II is modified from this menu. The SETUP sub-menu is protected with a password that must be entered the first time the SETUP or diagnostics menus are entered. Once entered correctly, the password does not have to be entered again so long as the password or user level are not changed when in the EDIT menu.
{}

{Setup Edit}
{2.1}
\hSetup Edit\h

Edit all setup parameters. When the settings are changed, RobComm-II will use these new values. Unless the SAVE selection is made, RobComm-II will not remember these values the next time.
{}

{Communications}
{2.1.1}
\hCommunications Parameter Block Editing\h

This entry block gives the user access to all the user-modifiable system parameters dealing with the communication to the robot.

See also the \mEnvironment\m parameter block.
{}

{ACI Baud}
{2.1.1.1}
\hACI Baud\h

When ACI file \mtransfers\m are taking place, this defines the baud rate of the serial communication. Standard selections can be made up to 19200 baud. See the \mPACING\m function for more information.
{}

{PACING}
{2.1.1.2}
\hPacing\h

The PACING value is a time in milliseconds that determines the pause between characters that are sent from the IBM-PC. A delay is typically

required for any communication above 2400 baud, since the robot controller treats serial communication as a low priority. The pacing value is not performed when the robot controller transmits information, so essentially the pacing function acts to provide a dual speed communication, and full speed is realized when sending data from the robot to the PC.
    \mACI Baud\m
{}

{2.1.1.3}
\hTerminal Baud\h

Determines the baud rate of communication during terminal emulation mode.
{}

{Change Slave}
{Slave Controller}
{2.1.1.4}
\hController Slave Number\h

The ACI link for file transfer has a multi-drop capability. Each robot on the link must have a slave number between 1 and 127. When using single channel mode it is important that this slave I.D. is set up properly.
{}

{Single Channel}
{2.1.1.5}
\hSingle Channel Mode\h

Single channel mode refers to a mode where terminal and ACI file transfer can occur through one physical channel. Users of the old RobComm will recall the switch box which was used to convert a single channel from the back of the PC to the two channels at the back of the robot. This was necessary since \mACI\m and terminal mode on the robot controller had to be executed on separate channels prior to RAPL-II version 1.02. Now, when single channel is selected, all serial communications can occur over a single channel, and it is connected to DEVICE 1 at the rear of the controller. Using the \mPORT\m selection, the appropriate IBM adaptor can be selected. When single channel is selected, it is suggested that the jumper J11 be removed from the motherboard.

This will permit DEVICE 1 to be the power up terminal. Also, when single channel is selected, the ACI BAUD rate as selected in this menu will act for both modes.

Ideosynchracies:
   a) If the controller is operating with terminal
   on DEVICE 0, then when going into terminal mode
   for the first time, the ABORT button must be
   pressed on the teach pendant. or the controller
   must be turned off and on again. This is
   because the controller is in a tight loop,
   awaiting characters from DEVICE 0, and the
   controller must be broken from this loop.

   b) A TEACH START will bring the controller up
   in the default baud rates of each channel, 9600
   for DEVICE 0 and 2400 for DEVICE 1.

   c) The single channel mode uses special ACI
   command codes to switch between terminal mode
   and ACI mode. For this reason, the single
   channel mode is best suited for multi-drop
   applications where a destination controller
   must select itself on the serial bus by raising
   its transmit enable signal. This remains in
   effect until the terminal mode is ended, after
   which the transmit enable turns off, and the
   controller returns back to the dormant ACI mode
   Failure for the ACI code to reach the
   controller may mean that the terminal mode will
   not be entered.
{}

{Communication Port}
{port}
{2.1.1.6}
                \hPort\h

The RobComm-II system can communicate either
through COM1 or COM2 or both standard IBM
channels. When using a single port, then this
selection will determine which port is used.
When both channels are selected, then the ACI
channel is connected to COM2, and the terminal
channel is connected to COM1. Note that this
does not determine how the "single channel" mode
works. \mSingle Channel\m Mode refers to the number
of serial channels that are used by the robot
controller.
{}

{Environment}
{2.1.2}
      \hEnvironment Parameter Block Editing\h

This entry block gives the user access to all the user-modifiable system parameters dealing with the Robcomm-II environment.

See also the \mCommunications\m parameter block.
{}

{On Collision}
{2.1.2.1}
\hOn Collision\h

This selection provides a default action for any table entry collisions as described in \mCollision Menu\m. Either Overwrite, Ignore or Rename can be automatically selected, as well as Query (default) which will utilize the collision menu at each collision.

With the Rename selection, the user will be prompted to enter a new name for the entry.
{}

{2.1.2.2} \hTerminal Log\h

All terminal interaction can be logged to a text file TERMINAL.LOG, that is in the RobComm-II main disk directory. This is useful when debugging, as it maintains a record of all user access with the robot.

The Log function is selected as a toggle from the setup menu. Hitting the ENTER key will turn it OFF or ON.

All data that is sent to the log file is appended, with a date and time that the session started and stopped. {}

{Use Defaults}
{Default}
{2.1.2.3}
\hUse Defaults\h

This selection is used to determine how the file handling utilities operate. These utilities are found in the FILE sub-menu. \mTOKENIZE\m, \mUNTOKENIZE\m, \mRENUMBER\m will all prompt for some types of arguments before they execute. Typically they ask for a destination file name for the operation. When the DEFAULTS is set to ON, then the

destination file is the source file name, and the source file will be given the .BAK extension. This mode is useful when processing a lot of files using the "\mALL BUT\m" selection in the file selection menu. This way, all files will be processed without any user intervention.
{}

{2.1.2.4}
{Extend Job Save}
\hExtended Job Save\h

This is a ON / OFF flag which tells Robcomm-II whether or not to prompt for PCP and HEX file information when saving jobs. When ON, this information will be prompted for. When OFF, it will not.

See also \mJob Send\m
{}

{2.1.2.5}
{Select All}
\hSelect All\h

This flag, when ON, will simulate an \mAll But\m selection during a \mMultiple Entry\m transfer. The menu will never be displayed and the application will automatically proceed with the processing of all entries. This is convenient when saving jobs.
{}

{Password}
{2.1.2.6}
\hPassword\h

A Password must be entered to get into the diagnostics functions, or to get into any level of the \mSETUP\m sub-menu. This password will protect against any unauthorized entry. If the password is changed, then the user must re-enter the password the next time the SETUP menu is entered.

See also \mPassword Check Enable\m
{}

{2.1.2.7}
{Auto Allocation} \hAuto Allocation\h

Job files can detect and automatically allocate robot memory according to the allocation information contained in the job file. When this flag is ON,

this auto allocation will take place without user intervention. If the flag is OFF then the user is informed of the failure and a screen of memory size available, requested and minimum is generated for comparison. If the current robot allocation is different from desired but the desired allocation will fit into the robot then no change is made to the current allocation.

See also \mJob Send\m{}

{2.1.2.8}
{Terminal Macro Keys}
\hTerminal Macro Keys\h

For keeping compatibility with old versions of Robcomm, the terminal emulator has command macro expansions mapped to the ALT-F1 through ALT-F10 keys. These macros are defined in the files F1.KEY to F10.KEY. The searching for these files and subsequent loading can be turned off with this flag to save time and effort if they are not present. When the flag is ON, Robcomm-II will search for the files each time it enters the emulator until it finds and loads them. When the flag is OFF, Robcomm-II will neither search for nor load the files.{}

{2.1.2.9}
{Starting Line}
\hRenumber Starting Line\h

This cell specifies and allows the user to change the line number that Robcomm-II starts at when renumbering a program file. This is the value used by the renumber routine when \mUse Defaults\m is set ON.

When Use Defaults is OFF, the renumber routine will still prompt for this entry and any changes at that point will be reflected in this cell.

See also \mRenumber\m
\mLine Increment\m
{}

{2.1.2.11}
{Line Increment}
\hRenumber Line Increment\h

This cell specifies and allows the user to change the line increment that Robcomm-II uses when renumbering a program file. This is the value used by the

renumber routine when \mUse Defaults\m is set ON.

When Use Defaults is OFF, the renumber routine will still prompt for this entry and any changes at that point will be reflected in this cell.

See also \mRenumber\m
\mStarting Line\m
{}

{User Level}
{2.1.2.10}
\hUser Level\h

The User Level selection permits the system supervisor to select a level with which the user can interact with RAPL-II. RAPL-II version 1.03 and later offers the flexibility to permit access to commands in one of four levels. These levels range from Novice to Supervisor.

This function is useful in an Educational setting where access to many high level functions is not necessary or confusing to the operator.
{}

{Direct Video}
{2.1.2.12}
\hVideo\h

Some IBM-PC clones do not have completely compatible BIOS display support. When the video is ON, then this will not utilize the computer's BIOS to perform screen handling, Rather it will access video memory directly. Typically this is faster. If set to OFF, then all screen access is performed through the BIOS. Select whichever works best on your machine, and then leave this alone.
{}

{2.1.2.13}
\hCommand File\h

A command file can be selected for the \mTOKENIZE\m, \mUNTOKENIZE\m and \mRENUMBER\m utilities. It is this file which defines the command strings for each TOKEN number. Language translation can be performed at this level, by replacing the 8 character command strings. The order of the strings must be maintained. RobComm-II comes

standard with the English ENGLISH.CMD and
RAPL2.CMD files as well as the German command
set GERMAN.CMD.

Notes:
1 German and English are supported only at the
  RAPL-II interpreter level.
2 The command file must be located in the
  RobComm-II main directory.
{}

{Colour}
{2.1.2.14}
                    \hColour\h

The processing of the colour windows as defined
in the ROBCOMM.MNU file will be used. When the
colour selection is changed, the change must be
SAVED, and then RobComm-II must be restarted from
the DOS prompt.
{}

{Text Editor}
{2.1.2.15}
                \hEditor\h

The user can define his or her favourite text
editor. This menu selection defines the complete
path name for the editor. The edit file specified
in the file selection window will be appended
onto this string to form a complete DOS command.
{}

{Shell Commands}
{2.1.3.1}
{2.1.3.2}
{2.1.3.3}
{2.1.3.4}
{2.1.3.5}
{2.1.3.6}
{2.1.3.7}
{2.1.3.8}
{2.1.3.9}
{2.1.3.10}
{2.1.3}   \h<Cntrl>Fx DOS Command String Editing\h

This gives the user access to the DOS command strings
associated with the control-function keys 1 through 10.
Each is fully modifiable and initialized from the file
FUNCTION.KEY. See the Robcomm-II manual for details

on the format of the file and strings in the file.
Any desired command line parameters may be included with
the command itself.

Two control characters can appear as the first characters
of any DOS sequence:

The control character \h~\h in the line will cause
Robcomm-II to prompt for command line arguments.

The control character \h^\h in the line will cause
Robcomm-II to purge from memory before the DOS function
is loaded. This is done in order to load large external
applications. When the application finishes, then
ROBCOMM-II is re-loaded. This process is done in the
following stages:
1 ROBCOMM-II is purged from memory and as that is being
   done TSR.COM is loaded into memory.
2 TSR.COM is executed, with the DOS application as an
  argument string.
3 The application is executed, and then terminates.
4 TSR.COM kills itself, but invokes ROBCOMM-II again.

The only drawback of this control mode, it that it takes
time to reload ROBCOMM-II. Use this control code with
discretion.

See also \mArgument Line\m
{}

{2.2}
\hSetup Save\h

Save the current settings of all system
parameters into the file ROBCOMM.CFG, which will
reside in the directory in which RobComm-II is
located.
{}

{2.3}
\hSetup Load\h

Re-load the settings defined in the ROBCOMM.CFG
file that resides in the same directory that
ROBCOMM.EXE resides in. This is useful if the
settings have been changed and the defaults have
to be restored.
{}

{2.3.2}
\hSetup file\h

Setup information can be read from the default file ROBCOMM.CFG, or another source can be selected from this window.
{}

{Password Check Enable}
{2.4}
\hPassword Check Enable\h
Password checking can be enabled for all restricted access areas of RobComm. Normally, once the operator has entered the password, unlimited access to RobComm is possible. If a user wishes to enable the password checking after he has already had access, then hitting this selection will force the password to be re-entered the next time a limited access item is selected.
{}

{3}
\hMemory Selection\h

This feature permits the user to deal directly with the robot memory. This is useful for two purposes:

1) to read or send new position calibration numbers, or
2) to diagnose potential problems by examining the robot memory

{}

{3.2}
\hCalibrate\h

The robot arm axes have calibration values which are loaded into the position registers when a HOME is executed. These values can be saved on disk in the event of a memory crash that could destroy them. Saving these values will eliminate the need to execute the entire CALIBRATE process. In other cases, new calibration values can be saved for different HOMING positions of the arm. For instance, the arm is always calibrated at the READY position from the factory. Special applications may mean that the robot is calibrated from homing brackets or other positions, in which case the factory values are not valid. Saving different calibration values for different applications is then made easy.
{}

{3.2.1}
\hSend Calibration Values\h

Re-load the calibration data values from disk to the robot. A HOME can be executed and the robot will be ready for action. A file name is selected which will be the source of the calibration data. All files with a .CAL extension will be listed, and only one can be selected.
{}

{3.2.1.1}
Select Calibration File to Send to Robot Controller

Select any of the existing calibration files in this directory. Only one can be selected.
{}

{3.2.2}
\hReceive Calibration Values\h

Save the calibration values from the robot memory to disk. A file name is entered. Typically, the name is SRS_NNNN.CAL, where NNNN is the serial number of the robot when shipped from the factory. The file name is built with a "SRS_" prefix which can be changed. For instance, if a file SRS_2001.CAL is present for robot serial number 2001, then if a new set of calibration values is created for use with a homing bracket, then a file name of BRK_2001.CAL may be appropriate, Never save new values under the old file name, as you will then loose the factory values.
{}

{3.2.2.1}
\hEnter Calibration File name for Calibration Save\h

Enter the four digit robot serial number, or re-enter the complete file name up to a maximum of 8 characters.
{}

{3.1}
\hDiagnostics\h

Robot memory diagnostics is important, as it
allows you to determine causes of some types
of failures.
{}

{3.1.3}
\hSegment\h

The memory EDIT function uses this segment value
to access the user memory. A default selection
of 0 should not be changed.
{}

{3.1.2}
\hEdit Memory\h

Not to be confused with file EDIT, the edit
screen allows you to process pages of raw robot
memory data. This may be useful when using PCPs
to run a process. The robot memory can be
displayed and manipulated in a number of formats
and data sizes.

\mTables\m
\mStandard\m
\mParameter\m
{}

{Table Editing}
{3.1.2.2}
\hRobot Table Editing\h

This selection allows the user to access and
modify the variable and location tables.
Both names and values can be modified.
As well, the position of entries can be modified
through the Sort and Move features of the editor
{}

{3.1.2.2.1}
\hVariable Table Editing\h

This allows the user to edit entries in the variable
table, including renaming, deleting, and moving
individual entries and sorting the table.

\mMove\m
\mSort\m
{}

{3.1.2.2.2}
\hLocation Table Editing\h

This allows the user to edit entries in the location table, including renaming, deleting, and moving individual entries and sorting the entire table.

\mMove\m
\mSort\m
{}

{Sort}
\hSorting Tables (ALT-S)\h

Both the variable and location tables can be sorted. The variable table can be sorted by either the variable \hName\h or the variable \hValue\h. Locations can only be sorted by Name.

In addition the sort order can be chosen, either \hAscending\h or \hDescending\h.
\mMove\m \mSort Key\m \mSort Direction\m
{}

{Sort Key}
\hSort Key Selection\h

When sorting variable table entries, the table can be sorted by either the name or the variable value.
{}

{Sort Direction}
\hSort Direction\h

When sorting both variable and location tables, the direction of the sort can be chosen, either \hAscending\h or \hDescending\h.

When sorting location tables, precision points are found lexically above cartesian locations. They are then found at the top of the table sorting in ascending order and at the bottom of the table in descending order.
{}

{Move}
\hMoving Table Entries (ALT-M)\h

Any entry in either of the tables can be moved by first cursoring to the desired entry and then pressing ALT-M.

The entire entry will be highlighted and the entry can be moved to the desired location. UP, DOWN, PAGE-UP and

PAGE-DOWN work as before, LEFT and RIGHT are non-functional.

When the entry is in the desired location, pressing ENTER will end the move and normal editing can resume.

\mSort\m
{}

{3.1.2.1}
{Standard Editing}
\hStandard Type Editing\h

The Standard Type Editing gives the user unlimited access to the robots memory and should be used with much caution. Any of 8 different basic types can be selected, with some of these having different display options as well. Each editing screen holds 256 bytes of robot memory in all standard modes.

\mByte and Char\m \mWord and Integer\m
\mLong Values\m \mFloating Point\m
\mPointers\m {}

{3.1.2.1.1}
{3.1.2.1.2}
{3.1.2.3.1}
{3.1.2.3.2}
{Byte and Char}
\hByte and Char Editing\h

In these modes, the data requested (either raw data or from the parameter table) is displayed in both byte and character format at the same time. Switching between the two formats is done with the TAB and SHIFT-TAB keys. Individual bytes are the atomic size for these editing formats.

\mUnsigned Displays\m
{}

{Unsigned Displays}
{3.1.2.1.1.1}
{3.1.2.1.1.2}
{3.1.2.3.1.1}
{3.1.2.3.1.2}
{3.1.2.1.3.1}
{3.1.2.1.3.2}
{3.1.2.3.3.1}
{3.1.2.3.3.2}
{3.1.2.1.5.1}
{3.1.2.1.5.2}

```
{3.1.2.3.5.1}
{3.1.2.3.5.2}
             \hUnsigned Decimal Displays\h

All unsigned decimal editing formats can be displayed
and modified in either decimal or hex notation.

            \mByte and Char\m
            \mWord and Integer\m
            \mLong Values\m
{}

{3.1.2.1.3}
{3.1.2.1.4}
{3.1.2.3.3}
{3.1.2.3.4}
{Word and Integer}
             \hWord and Integer Editing\h

In this editing mode, the atomic size is two bytes.
With the Word format, either HEX or unsigned
decimal display can be utilized, while integer
displays in signed decimal.

            \mUnsigned Displays\m
{}

{3.1.2.1.5}
{3.1.2.1.6}
{3.1.2.3.5}
{3.1.2.3.6}
{Long Values}
        \hLong Signed and Unsigned Editing\h

The atomic size for these modes is 4 bytes.  With
the Unsigned format, the display can be in either
HEX or decimal format.  The signed display is in
decimal only.

            \mUnsigned Displays\m
{}

{3.1.2.1.7}
{3.1.2.3.7}
{Floating Point}
           \hFloating Point Editing\h

This selection allows the user to access memory
in floating point format.  Two types of format are
available; \mFixed Point\m and \mScientific Notation\m
The atomic size in both cases is 4 bytes.
{}
```

```
{3.1.2.1.7.1}
{3.1.2.3.7.1}
{Fixed Point}
          \hFixed Point Editing\h

This displays the data with a fixed number of
digits after the decimal point.  Numbers
which can not be displayed in the fixed point
format are displayed in the scientific notation
format.
          \mScientific Notation\m
          \mFloating Point\m
{}

{3.1.2.1.7.2}
{3.1.2.3.7.2}
{Scientific Notation}
         \hScientific Notation Editing\h

This displays the data in scientific or engineering
notation with a mantissa of significant bits and
an exponential multiplier to reach the proper
magnitude.

         \mFixed Point Editing\m
         \mFloating Point\m
{}

{3.1.2.1.8}
{3.1.2.3.8}
{Pointers}
             \hPointer Editing\h

Allows the user to edit the memory in the standard
seg:offs format.  Atomic size for this editing
format is 4 bytes.
{}

{3.1.2.3}
            \hParameter List Editing\h

With this selection, the user has direct access to
the various RAPL-II parameter lists, along with
descriptions for each parameter.  Users should be
familiar with RAPL-II and robot controller
operations before modifying these parameters.

  \mByte and Char\m       \mWord and Integer\m
  \mLong Values\m         \mFloating Point\m
  \mPointers\m {}
```

```
{3.1.1}
            \hView Memory Pointers\h

A pointer list of root memory parameters is
provided in this window.
{}

{PFF File Transfer}
{3.1.4}
          \hPFF File transfer\h

A PFF file (parameter file format) is a data file
which contains ASCII representation of raw memory
data.
     \mPFF Load\m
     \mPFF Save\m
{}

{PFF Load}
{3.1.4.1}
           \hSend PFF file\h

     \mPFF File Transfer\m
{}

{PFF Save}
{3.1.4.2}
          \hReceive PFF file\h

     \mPFF File Transfer\m
{}

{Terminal Emulation}
{4}
         \hTerminal Emulation Mode\h

Depending upon the setting of the \mSingle Channel\m
mode in the SETUP EDIT screen, entering this
selection will send you to the emulation window
either immediately, or after the system prompts
you to select the terminal setting on the channel
selector switch.

See also \mTerminal Macro Keys\m
{}

{Collision Menu}
          \hCollision Menu Selection\h

During some form of table transfer to the robot, a
duplicate table entry was found ( a name in the file
matches a name in the actual table of the robot ).
```

In this instance, some form of collision resolution is required. The user may overwrite the current robot entry with the file entry, ignore or rename the file entry or abort the transmission.

An automatic selection for this menu may be made with the \mOn Collision\m parameter in the environment configuration block.
{}

{Edit File}
\hEdit File Entry\h

Most editors today support specifying the name of the file to edit on the command line. In support of this, this entry can be used to specify the file name to edit. It will then be included in the invocation of the editor as the first command line parameter.
{}

{Affirmation}
\hAffirmation Menu Selection\h

At this point, some question requires a YES or NO response. This menu allows you to select either YES or NO in response to the question.
{}

{Destination File}
\hDestination File Entry\h

This entry is used to specify the name of the file to generate during the given application. For example, this file name is the file created (with a .VAR extension) during a variable save.
{}

{Argument Line}
\hArgument Line Entry\h

Robcomm-II allows the CNTRL-F1 through CNTRL-F10 to be mapped to user definable DOS commands. The format of these command strings allows the user to specify the input of a command line to be appended to the actual command. In addition, any un-defined key strings default to a command entry. This entry is used to enter in the command line arguments for a defined command or the command itself (with arguments) for an undefined command string.

To access this argument line entry, the control code

\h~\h must be the first character of the command string. Any parameters included with the command string in the definition are accessible with the argument line entry.

See \mShell Commands\m for information on (re)defining the function keys.
{}

{Multiple Entry}
{Program Selection}
\hMultiple Entry Selection\h

When RobComm-II provides a directory of entries for a particular operation, the user typically has the ability to select several entries at once. This is done by using the arrow keys to position the highlighted bar above the choice, and then by using the ENTER or SPACE key, the item will be selected. This selection is shown by a highlighted icon to the left of the file name.

If all entries are to be selected, then by simply selecting the first item name "\mALL BUT\m" and nothing else, all entries will be selected. If any entry is selected along with "\mALL BUT\m", that entry will be ignored from the processing.

Whenever RobComm-II displays a directory of files, it assumes one of the standard RobComm-II \mFile Types\m.

A \mQuick Search\m mask is available to user to select a group of entries based on the entry name.

See also the setting of the flag \mSelect All\m in the environment configuration setup.
{}

{Quick Search}
\hQuick Search Mask\h
Entering a name (or partial name) and pressing <CR> will select the first file which matches the mask. All files which match the mask can be selected by adding the wild card character \h*\h to the end of the mask. As the mask becomes more complex (longer) the current position changes to reflect the first entry which matches the mask.

A mask consisting of a \h*\h alone is equivalent to selecting the \mAll But\m entry (if available)
{}

```
{ALL BUT}
          \hALL BUT...\h
 When RobComm-II permits multiple entry selections,
 then the ALL BUT selection can be used to select
 all files.
 When the ALL BUT is selected, then the entries
 that are selected along with it will not be
 processed.

        \mMultiple Entry\m
         \mSelect All\m
         \mQuick Search\m
{}


{File Selection}
          \hSingle File Selection\h

 Single file selection is used whenever only one
 file can be selected for the next operation. This
 is found for a \mJob Send\m, a \mHEX file Send\m,
 \mPCP Load\m or Variable and Location file loads.
 The \mQuick Search entry is active for this file
 selection for increased ease of finding the
 desired file.

 \mFile Types\m.

{}


{File Types}
    \hDescription of RobComm-II File Types\h

 RobComm-II uses a variety of file types to store
 specific information. Here is a list:
 .ROB RAPL-II program files. These are text files
      that always end with a '$' character. These
      files can be read into the robot memory, or
      they can be received from the robot memory
      for storage or processing
      \mProgram Send\m
      \mProgram Receive\m
      \mTokenize\m
      \mUntokenize\m
      \mRenumber\m
 .VAR
      VAR files contain the RAPL variable information.
      The VAR file will be created when reading infor-
      mation out of the robot when performing a
```

variable save or a job save.
\mVariable Send\m
\mVariable Receive\m

.LOC

\mLocation Send\m
\mLocation Receive\m

.HEX

HEX files can be created when reading memory data out of the robot controller, or when performing a PCP load. The HEX file creation here is the first step towards creating a PCP loadable file from the EXE parent file.
\mHEX Send\m
\mHEX Receive\m

.EXE

The only time an EXE file is used as a data file, it is the source for a PCP load operation. There are other EXE files on the RobComm-II distribution diskette, but these are the system files, and they should not be modified at all. All of the system executables are:
ROBCOMM.EXE COMPILE.EXE
HELPINDX.EXE

\mPCP Load\m

.HLP

The ROBCOMM.HLP file is the text file which contains all of the ROBCOMM.HLP help information. In this file, all indexed items are bounded by brace brackets { and }. The text that follows the indexed item will be displayed when that item is selected. Phrases bounded by '\\m' are menu items which will be highlighted when that page of text is shown. The cursor keys allow you to automatically position on top of these items and selecting that highlighted item will then step you to the associated text, if that item has an index.
If the .HLP file has been changed, it must be recompiled using the HELPINDX.EXE compiler. This will create a ROBCOMM.HDX, which indexes the help data for RobComm-II.

.MNU

The ROBCOMM.MNU file is the text menu file that defines all of RobComm-II's operating menu characteristics. It can be changed with any text editor. If it has been changed, then the COMPILE.EXE menu compiler will be called, and it will create a ROBCOMM.MNC compiled menu file which is actually used by RobComm-II.

.PFF

```
      A parameter file format file (PFF) is a text
      file which contains information about RAPL-II
      system parameters. Using this file format, a
      programmer can determine what parameters are
      to be loaded and saved. This permits the
      construction of advanced user interfaces to
      RAPL-II by third party software sources.
      \mPFF Save\m
      \mPFF Load\m
 .HDX
 .CAL

      \mProgram Selection\m
{}

{UTILITY ERRORS}
       \hRobcomm-II Utility Errors  0\h\rN\r\h00\h
These errors are generated by the various functions in
Robcomm-II and are related to file and disk access.

The second digit of the error number indicates which
of 10 utility error codes has been generated.

The \rN\r can be as follows:

   1  Memory Allocation
   2  File Read Error     7  Data Modify Not Allowed
   3  File Write Error    8  Application Failure
   4  File Format Error   9  File Find Error
   5  File Create Error   A  File Data Error

See Also \mACI Errors\m
{}

{ERROR 0001}
{Short Answer}
Ignore this error and continue into terminal
emulation. When in the emulator, exit using
the F10 key, then re-enter. The error should
go away. If you want more information then:
        \mLong Answer\m
{}

{Long Answer}
Error 0001 can occur when attempting to enter
terminal emulation when in single channel mode.
In single channel mode, the controller's serial
device #1 flip-flops between ASCII communication
and the ACI protocol. When ROBCOMM attempts to
enter emulation mode, it shuts off the ACI
protocol with a special ACI transmission. IF the
controller is not in ACI mode, then this communication
```

will fail, which means that this message appears. When
you enter the terminal screen, operation should proceed
normally. If this doesn't happen, then see:
\mWrong Baud Rate\m
\mShort Answer\m
{}

{Wrong Baud Rate}
Error 0001 can occur when attempting to enter
terminal emulation when in single channel mode, when the
baud rates of the controller and PC do not match. It is
quite common to see this error when changing the controller
and PC baud rates to new values. When changing baud rates,
the controller is changed first, using the interactive
\hCONFIG\h RAPL command. The terminal emulator is then
terminated. When this happens, the normal close-out message
that the PC sends to the controller will not longer succeed,
so the controller stays in terminal mode. The baud rate is
then changed in the PC. When terminal mode is re-entered, then
the scenario defined in \mLong Answer\m is made apparent.
{}

{ERROR 0100}
\hMemory Allocation Error\h

This error happens when there is problem when
attempting to allocate memory or open a file.

It may terminate the program depending on where
in the package it occurs.

Often it will only terminate the utility in progress.

This is a sign that there is not enough memory free in
the system to properly run Robcomm. Robcomm requires
at least 256k bytes of free memory, while the maximum
used is 350k bytes.

See Also \mUtility Errors\m
{}

{ERROR 0200}
\hFile Read Error\h

This error occurs when a problem arises while trying
to read from a file.

Check the disk the file is being read from to ensure
it is ready for the transmission.
Check also to ensure the file in question is present
on the disk, in the directory being searched.

See Also \mUtility Errors\m
{}

{ERROR 0300}
\hFile Write Error\h

This error occurs when a problem arises while trying to write to a file.

Check to ensure the disk is ready to by written to. Check the disk to ensure there is enough room for the file in question.

See Also \mUtility Errors\m
{}

{ERROR 0400}
\hFile Format Error\h

Many Robcomm-II utilities require files with a certain format. This error indicates an application was trying to read a file with a format other than what it was expecting.

Check the file to ensure it has the proper format for operations being performed on it.

See Also \mUtility Errors\m
{}

{ERROR 0500}
\hFile Creation Error\h

This error pops up when the creation of a file is blocked for some reason by DOS.

Reasons are multiple. The file could already exist or could be marked as read only to prevent overwriting. Another reason could be that the disk or directory is full and can not accept more files.

Check the disk and directory in question for room as well as the attributes of the file.

See Also \mUtility Errors\m
{}

{ERROR 0700}
\hData Modify Not Allowed\h

An attempt has been made to open the file in question either for overwrite or for append. The problem is that

the file currently on disk has a Read Only attribute which prevents any modification to the existing file.

To get around this, simply remove the Read Only attribute. (see the DOS manual for instructions on changing attributes).

See Also \mUtility Errors\m
{}

{ERROR 0800}
\hApplication Failure\h

This is a general catch-all error code for errors which are one of a kind or very specific and don't quite fit into one of the pre-defined categories.

It means simply that something has happened which prevents the utility from completing. For more information on the specific error, carefully read the message in the Error window and if still in doubt, refer to the manual under the specific utility in which the error occurred.

See Also \mUtility Errors\m
{}

{ERROR 0900}
\hFile Find Error\h

This simply means the file in question was not found where the Robcomm utility was looking.

Check the disk and directory to ensure the file in question is where it is supposed to be.

Check the current directory to ensure it is pointing to the correct disk and directory.

See Also \mUtility Errors\m
{}

{ERROR 0A00}
\hFile Data Error\h

This error is very similar to the File Format Error (#0400) except it also indicates an unexpected end of file has occurred and opening zero length files for read operations.

See Also \mUtility Errors\m{}

{ACI ERRORS}
\hACI Error Codes 00\h\rNN\r

The ACI error codes indicate a problem in a transmission either to or from the robot.

Possible values for \rNN\r are as follows:

  04 \menquiry timeout\m
  12 \mheader transmit\m timeout
  16 \mimproper response\m from slave
  20 \mdata block receive\m timeout
  22 \mdata block transmit\m timeout
  24 \mexpecting NAK\m
  27 \mexpecting EOT\m
  28 \mexpecting STX\m
  32 \mexpecting ETB\m
  34 \mexpecting ETX\m
  40 general \mACI receive timeout\m
  50 general \mACI transmit timeout\m

See Also \mUtility Errors\m{}

{ERROR 0004}
{ENQUIRY TIMEOUT}
A timeout has occurred while waiting for the robot to respond to an ENQuiry code.

Check connections and robot power.

See Also \mACI Errors\m
{}

{ERROR 0016}
{IMPROPER RESPONSE}
{ERROR 0024}
{EXPECTING NAK}
{ERROR 0027}
{EXPECTING EOT}
{ERROR 0028}
{EXPECTING STX}
{ERROR 0032}
{EXPECTING ETB}
{ERROR 0034}
{EXPECTING ETX}
\hImproper Response from Slave\h

This error occurs when the Robcomm-II package expects to see certain control codes from the robot. These codes are used to regulate and control the ACI transmission.

Failure of these codes to appear as expected indicates some sort of transmission problem or data corruption problem. These errors are rarely seen

The best course of action is to check all serial connections and cables and retry the transmission.

See Also \mACI Errors\m{}

{ERROR 0020}
{DATA BLOCK RECEIVE}
{ERROR 0040}
{ACI RECEIVE TIMEOUT}
\hACI Receive Timeout\h

This indicates an over long wait for the receipt of characters through the ACI port.

Check cables connections to be sure they are tight and going to the proper serial outlets.

Check switch box if being used. It should be switched for device 1.

Check baud rates on both the PC and controller. They must be the same for the transfer to occur

See Also \mACI Errors\m
{}

{ERROR 0012}
{HEADER TRANSMIT}
{ERROR 0022}
{DATA BLOCK TRANSMIT}
{ERROR 0050}
{ACI TRANSMIT TIMEOUT}
\hACI Transmit Timeout\h

This indicates an over long wait for the robot to signal ready to receive characters during an ACI transmission.

Check cables connections to be sure they are tight and going to the proper serial outlets.

Check switch box if being used. It should be switched for device 1.

Check baud rates on both the PC and controller. They must be the same for the transfer to occur

See Also \mACI Errors\m

{}

{ERROR 1001}
\hInvalid password entered\h

A proper password must be entered to gain entrance into either the Setup/Edit menu or the Memory menu. This is to prevent unauthorized manipulation of low-level system parameters.
{}

{ERROR 1002}
\hPossible Robot Memory Corruption\h

The robot program memory may have been corrupted. The safest bet is to re-initialize the program table using the INIT command

This is usually the result of instable communications. To remedy the situation, try lowering the baud rate.
{}

{ERROR 1003}
\hLocation Table Full\h

There is no more room in the location table in which to put more locations.

One course of action is to remove locations from the table which are not needed.

Another, although more drastic, is to reallocate memory to provide enough room in to location table for all of the locations required.
{}

{ERROR 1004}
\hIncompatible RAPL Version\h

Some Robcomm-II functions are only available when used with certain versions of RAPL or RAPL-II.

For instance, the User Level feature is only available on RAPL-II versions 1.03 and 1.11 and above.

The number in the Error window is the version of RAPL running in the controller.
{}

{GENERAL ERROR}
This error has returned no error code

```
No help is available
{}

{CHECKSUM FAILURE}
This error is generated during variable and location table
loads or saves.  It indicates that the checksum in the
file or table does not match the newly calculated value
based on the entries current values.  This discrepency
informs the user that the file or table could have been
corrupted.  The possible actions are to \hDISCARD\h the
entry in which case the entry will not be transmitted,
\hRECALCULATE\h the checksum which will transmit the
entry with a recalculated checksum, or to
\hPASS UNTOUCHED\h which will transmit the entry as is.

DISCARD is the recommended selection unless the nature
of the error is known.
{}

{HARDWARE ERROR}
This error is generated when DOS reports some form of
hardware error ROBCOMM-II.  One of the standard DOS
\rABORT\r \rRETRY\r and \rIGNORE\r selections must be
made at this point.
{}

Revision History:
9/9/90 Added new items
10/11/90    Added ERROR items
01/14/91    Added advanced editing items including special
context help for
              sort direction and sort key menus
05/27/91     Added hardware error help
             Added checksum failure menu help

Notes:

\f special character for form feeds, which will cause a page
break
in the window display.
```