GOOSY
Id.: GPS
Version: 1.0
Date: 19–JAN–1994
Revised: January, 19 1994

GSI Online Offline System

# GOOSY PAW Server (GPS)

R.S. Mayer

January, 19 1994

GSI, Gesellschaft für Schwerionenforschung mbH
Postfach 11 05 52, Planckstraße 1, D-64220 Darmstadt
Tel. (0 6159) 71–0

# Chapter 1

# Preface

## 1.1 GOOSY Authors and Advisory Service

The authors of GOOSY and their main fields for advisory services are:

**M. Richter** GOOSY Data Management, VAX/VMS System Manager (Tel. 2394)

**R. Barth** GOOSY and PAW software (since 1995) (Tel. 2546)

**H.G. Essel** (GOOSY 1983-1993) Data Acquisition (Tel. 2491)

**N. Kurz** Data Acquisition (since 1992) (Tel. 2979)

**W. Ott** Data Acquisition (since 1994) (Tel. 2979)

People who have been involved in the development of GOOSY.

**B. Dechant** GOOSY software (1993-1095) (Tel. 2546)

**R. S. Mayer** Data Acquisition (1992-1995) (Tel. 2491)

**R. Fritzsche** Miscellanea (1989-1995) (Tel. 2419)

**H. Grein** Miscellanea (1984-1989)

**T. Kroll** Miscellanea, Printers (1984-1988)

**R. Thomitzek** Miscellanea, Printers, Terminals (1988-1989)

**W. Kynast** GIPSY preprocessor (1988)

**W.F.J. Müller** GOONET networking, Command interface (1984-1985)

**H. Sohlbach** J11, VME (1986-1989)

**W. Spreng** Display, Graphics (1984-1989)

**K. Winkelmann** GOOSY Data Elements, IBM (1984-1986)

## 1.2 Further GOOSY Manuals

The GOOSY system is described in the following manuals:

- GOOSY Introduction and Command Summary
- GOOSY Data Acquisition and Analysis
- GOOSY Data Management
- GOOSY Data Management Commands
- GOOSY Display
- GOOSY Hardware
- GOOSY DCL Procedures. GOOSY Error Recovery
- GOOSY Manual
- GOOSY Commands

Further manuals are available:

- GOOSY Buffer structures
- GOOSY PAW Server
- GOOSY LMD List Mode Data Generator
- SBS Single Branch System
- TCP-Package
- TRIGGER Bus
- VME Introduction
- OpenVMS Introduction

## 1.3 Intended Audience

This manual is written for GOOSY and PAW users. It assumes that the reader is familiar with most VAX-VMS concepts and commands and the environments (VMS, OpenVMS, UNIX-flavours) where PAW will be used. It provides all information necessary to use the GOOSY - PAW - Server. For VAX beginners the 'VMS Introduction' is recommended. For GOOSY beginners the 'GOOSY Introduction' is recommended.

Users of the PAW- and related software are kindfully asked to refer to the CERN software documentation. Our experience shows that updates in the software might result in different behavior. Please inform yourself about the current recommended version in your institute. Examples and templates for the use of the GPS client in a PAW environment are given here without any responsability of the GOOSY group and the user may **not** trust in regular updates.

## 1.4 Overview

- Section 2:
  GOOSY PAW Server: Introduction.
- Section 3:
  GPS GOOSY Server.
- Section 4:
  SBS Event Server.
- Section 5:
  GPS PAW Client.
- Section 6:
  GPS Client (standalone).

The author would be grateful for any critical comment or any suggestion about this manual.

## 1.5 GOOSY Copy Right

The GOOSY software package has been developed at GSI for scientific applications. Any distribution or usage of GOOSY without permission of GSI is not allowed. To get the permission, please contact Dr. M. Richter at GSI.
Registered Trademarks are not explicitly noted.

# Chapter 2

# Introduction

## 2.1 GOOSY - PAW - Server: Introduction

In order to allow online data acquisition control with other than GOOSY analysis software, the GPS tools have been developed. On the acquisition side

- The GPS-Server runs in the GOOSY Transport Manager environment.
- The SBS(Single Branch System) Event Server runs on the CVC (in preparation).

The GPS-Client(s) may run on one of the following computer platforms: VAX-VMS, AXP-OpenVMS, DEC-Ultrix, HP-UX, AIX (IBM). The client(s) may connect to the GOOSY server or SBS Event Server and request events. The client calls a user written analysis routine. Filters may be specified for the client to get selected events from the GOOSY server or SBS Event Server.

### 2.1.1 GPS GOOSY Server

The GPS Server can be started in a GOOSY environment with a running Transport Manager (see chapter 3).

The GOOSY PAW server may send event data from $TMR mailbox input to TCP/IP network clients. Serveral clients from the same or different hosts and systems may connect via TCP/IP and get data. The Clients may specify filter criteria, then only the selected event types are send to the client. Data input can be or from online data acquisition or from file input. When the server is started successfully, a port number is returned. This port number, together with the node name where the server runs, is required for the client to connect to the server.
The server tries to connect a port server and to communicate

- the name of the service, i.e. the full mailbox name used by the server as data input,
- the port number,
- the node name of the server and
- (a character string with information text).

Either the **service-name** (if not unique the **node::service-name**) when using the port server or **node** and **port** are required by the client.

### 2.1.2 SBS Event Server

The SBS Event Server (see chapter 4) is the equivalent part to the GPS server running on the CVC in the Single Branch System (SBS). The SBS Event Server get events from the data acquisition stream.

### 2.1.3 GPS PAW Client

This section (chapter 5) describes the full GPS client functionality.

The GPS-PAW-Client connect to a running Server (MGOOPS), send filter condition and request event data. After each event a user analysis routine is called (UANAL). The client connects to an existing server on a VMS node. To connect a client to a server, you need to specify the node and the port number. Alternatively and to simplificate the use, you may specify only the name of the server and port=0. (In case of ambiguity, i.e. several servers with the same name on different nodes, you must specify node::name). The client gets node and port from the port server and connects automatically.

Some hints are given how to build your own PAW when using the GPS client tool (see chapter 1.3). Until now, test have been done on VAX-VMS for the time being. When using the given command definition file

- an online help and
- an input menu

is available in the PAW environment.

### 2.1.4 GPS Client (standalone)

The GPS Client described in section 6 is a client test program. The functionality is the same as the GPS-PAW-Client routine, for the description refer also section 5. pc_client runs on VAX-VMS, AXP-OpenVMS and DEC-Ultrix. It will soon be available on AIX(IBM) and HP-UX.

### 2.1.5 Internals on GPS Server and Client

Internals on software modules used by the GPS Server and the GPS Client are available in the manual: **GOOSY PAW Server (GPS) Full Documentation** on VMS GOO$DOC:GM_GPS_FULL.PS. Please refer it when you intend to develope your own software using GPS.

# Chapter 3

# GPS GOOSY Server

## MGOOPS

| | |
|---|---|
| **CALLING** | GOOSY CRE PROC GPS $GPS |
| **PURPOSE** | GOOSY PAW server. Starts the GOOSY process $GPS |

## DESCRIPTION

| | |
|---|---|
| **CALLING** | GOOSY CRE PROC GPS $GPS |
| **FUNCTION** | GOOSY PAW server may send event data from $TMR mailbox input to TCP/IP network clients. See GOOSY command START SERVER. |

## IMPLEMENTATION

| | |
|---|---|
| **Return type** | BIN FIXED(31) |
| **Status codes** | Status of last command |
| **File name** | MGOOPS.PPL |
| **Version** | 1.01 |
| **Author** | H.G.Essel |
| **Last Update** | 28-Jul-1993 |

## INTERNALS

| | |
|---|---|
| **Utility** | IO |
| **Compile lib.** | GOOINC.TLB |
| **Home direct.** | GOO$IO |
| **Created** | 28-JUL-1993 |

## Updates

| **Update** | Date | Purpose |
|---|---|---|
| | **10-Sep-93** | Documentation (RSM) |
| | **01-Dec-93** | Delete name server entry (RSM) |

# MGOOPS_START

**CALLING** STS = MGOOPS_START(CV_envi,
L_channel,
I_clients,
I_sec)

**PURPOSE** Start GOOSY PAW server

# DESCRIPTION

**FUNCTION** Starts GOOSY PAW server, which takes event data from the specified GOOSY mailbox. The mailbox is served by the GOOSY TransportMangeR either from an online data acquisition or from LMD-file input.

Several clients may connect to the server and receive event data. The number of allowed clients can be specified (max=20, def=10). For client interna see R.S.Mayer (Tel: -491). For the PAW implementation on VMS and UNIX please contact H.Dbbeling.

For further information see also the keywords MGOOPS, I$PS_SERVER and I$PS_PROC. Stop the server with the GOOSY command SHUT SERVER.

## Command

**Command keys** START SERVER

**START SERVER EnvName Channel Clients Loginter**

**PURPOSE** Starts GOOSY PAW server, which takes event data from the specified GOOSY mailbox. Several clients may connect to the server and receive event data. See also MGOOPS.

**PARAMETERS**

**EnvName** Environment name (def=actual environment)

**Channel** Mailbox number 1,2 or 3 (def=1)

**Clients** Maximum number of clients allowed to connect (max=20, def=10)

| | |
|---|---|
| **Loginter** | Interval [sec]write status information about the server and the connected clients to SLOG-file. (def=60) |
| **Return type** | BIN FIXED(31) |
| **Status codes** | Status of last command |
| **File name** | MGOOPS.PPL |
| **Version** | 1.01 |
| **Author** | H.G.Essel |
| **Last Update** | 24-Aug-1993 (RSM) |

## UPDATES

| | |
|---|---|
| **Update** | Date Purpose |

# MGOOPS_STOP

**CALLING** STS = MGOOPS_STOP()

**PURPOSE** Shut down GOOSY PAW server

## DESCRIPTION

**FUNCTION** Shut down the GOOSY - PAW server, terminates the process $GPS. Delete name server entry.

## Command

**Command keys** SHUT SERVER

### SHUT SERVER

**PURPOSE** Shuts down the GOOSY PAW server, terminates the process $GPS

**Return type** BIN FIXED(31)

**Status codes** Status of last command

**File name** MGOOPS.PPL

**Version** 1.01

**Author** H.G.Essel

**Last Update** 24-Aug-1993 (RSM)

## UPDATES

**Updates** Date Purpose

**01-Dec-93** Delete name server entry (RSM)

# Chapter 4

# SBS Event Server

## 4.1 SBS Event Server

The connection and the event request is identical to the GOOSY GPS Server (see 3).

The SBS Event Server is available since May-94 and documented in the manual: **Single Branch System** on VMS GOO$DOC:GM_SBS.PS.

# Chapter 5

# GPS PAW Client

# F_CLIPAW

**f_clipaw(pc_hostn,pl_portn,pl_nbevt,pc_filtr,**
**pl_sampl,pl_echo,pc_tyevt,pl_bflsh)**

**PURPOSE** Client to connect GOOSY - PAW - Server or SBS - Event - Server. (See also HELP MGOOPS and documentation m_event_serv)

**ARGUMENTS** (see also Description Function)

**pc_hostn** (char *) Node_name

**pl_portn** (long *) Port_number > 0

**or** by using the port server

**pc_hostn** ptr to Server_name or Node_name::Server_name

**pl_portn** ptr to Port number <= 0

**pl_nbevt** (long *) Number of events

| | |
|---|---|
| **>0** | Requ number of events from the server |
| **=0** | Client will be started with the given parameters but no connection established |
| **-1** | Request an unlimited number of events |

**pc_filtr** (char *) Filter option or File name

| | |
|---|---|
| **A** | Select all events. |
| **F** | Filter condition will be prompted. |
| **file** | File with filter specifications. |

**pl_sampl** (long *) Reduction rate

| | |
|---|---|
| **0,1** | Get every event when filter match. |
| **n>0** | Every n'th event when filter match. |

**pl_echo** (long *) Echo rate

| | | |
|---|---|---|
| | **>0** | Notify after every n'th event. |
| | **0** | Disable echo. |
| | **n<0** | Notify every n percent of requ. events |

**pc_tyevt** (char *) Display option

| | |
|---|---|
| **X** | Switch off analysis routine UANAL |
| **N** | No display |
| **H** | Display only event (and subevt) Header |
| **D** | Display evt/sev Header and Data |

**pl_bflsh** (long *) Buffer flushing time (timeout = *10)

## Description

**FUNCTION** Connect to a running Server (GOOSY: MGOOPS or SBS: m_event_serv), send filter condition and request event data. After each event a user analysis routine is called (UANAL). The client connects to an existing server

| | |
|---|---|
| **GOOSY-GPS** | GOOSY GPS Server on a VAX-[Open]VMS node. The server can be started from a GOOSY environment with a running transport-manager ($TMR) process:<br>G>CREATE PROCESS GPS $GPS<br>G>START SERVER |
| **SBS-Event-Server** | m_event_serv on SBS (Single Branch System) |

**Description** (See VAX/VMS Help items MGOOPS, I$PS_SERVER and I$PS_PROC for further information). The starting server returns his port number and automatically reports his node, portnumber and name to a port server. Otherwise the Portnumber of the already running server must be known.

To connect a client to a server, you need to specify the node and the port number. Alternatively and to simplificate the use, you may specify only the name of the server and port=0. (In case of ambiguity, i.e. several servers with the same name on different nodes, you must specify node::name). The client gets node and port from the port server and connects automatically.

The client requests a number of events (or a continous stream) from the server. A strong selection on the events may be applied by specifiing filter conditions.

Samples of events that match the filter may be taken.

The reception of events may be echoed.

The display of the event (and subevent) header or the full data content may be selected. With the same parameter the user analysis UANAL may be switched on/off.

The server flushes the buffers to the client in a selectable time intervall. Even when the server gets no events from his input, or none of the events match the filter, the client gets a information buffer regularly.

Statistical information about the server and the client, i.e. read/written bytes, processed buffers, processed events, filter matching events etc. are containd in every buffer sent by the server.
CTRL_g invoces the call of a user modifiable (FORTRAN) subroutine UCLINFO that e.g. displays these information. (Response time on CTRL_g <= buffer_flushing_time)

CTRL_a terminates the client. (Response see above)

For every incomming event, the user written (FORTRAN) analysis routine UANAL is called. UANAL may also be skipped.

## Filter_description

**Filter criteria for event selection** Filters may be applied on the event (i.e. event header in the case of event type=10) and/or on subevents (if there are).

Several filter may be defined with logical conditions between them. Filter specifications may be grouped in so called blocks with logical conditions between them. Each block applies on a different region of the event (e.g. 1st block on event header, trigger number etc., 2nd block on subevent a etc. ...). In case of a subevent, one must unambigiously define in the first entry of the filter block on which subevent the block shall be applied (by giving e.g. the processor id).

**Output selection** The server may send the whole event that has fullfilled the filter criteria or only parts of it. This selection is independent of the filter. E.g. in the filter, the trigger number (event) and a pattern in the subevent b is checked, but only the subevent a and c will be send to the client. The output selection is the first entry of complete filter.

**Detailed filter description** In order to maintain software performance, all filter specifications (see topic) are internally translated into bit mask and offset. The mask and offset may be word or longword aligned. This makes necessary that some rules for the filter definitions have to be respected:

The required order is:

1. output selection
2. filter selection
   a) event specific filter
   b) subevent specific filter

Negligible disregard of these rules are automatically corrected and result in a warning message.

A complete filter consists of

1. block with the output selection and
2. one or more blocks with filter criteria.

In some cases, you need only one block for output selection and filter criteria (see topic examples).

Defining a block:
File input and interactive input are identical!

1. lenght of block, i.e. number of filter entries
2. filter entry(ies)

The filter specification consists of 9 entries
(see filter_specification).

1: Select filter or output selection for
event (1) or subevent (0).
2: Select filter specification (1)
else (0).
3: Select output specification (1)
else (0).
4: Operation code: see Filter_specification
ALL,IDENT,ANY,INCL,EXCL,LT,GE:
filter result = mask opcode object
(the 'object' is the word or longword
at the position 'offset' in the
event data)
5: Logical link between filter

specifications in a filter block:
0: OR 1: AND
6: Logical link between filter blocks:
0: OR 1: AND
7: Filter specification:
0 : Take all
1 : trigger
2 : pattern and offset
4 : type
8 : subtype
12: subtype & type *)
16: procid
32: contr & subcrate *)
48: contr & subcrate & procid *)
*) byte/word sequ. from left to right
8: Mask (=bit pattern)
9: Offset

## Filter_specification

### Filter specification

1.evtsev 2.selflt 3.selwrt 4.opc 5.lnkf1 6.lnkf2
7.fltspec 8.mask 9.offs

**1.** Select event/subevent

| | |
|---|---|
| **1** | event |
| **0** | subevent |

**2.** Select filter

| | |
|---|---|
| **0** | off |
| **1** | on |

**3.** Select write

| | |
|---|---|
| **0** | off |
| **1** | on |

4. Object code [Res = object opcode mask]

| | |
|---|---|
| **0** | !! (ALL) |
| **1** | == (IDENT) [object == mask] |

| | |
|---|---|
| **2** | && (ANY) [object & mask] |
| **3** | &= (INCL) [(object & mask) == object] |
| 4 | ^ = (EXCL) [(object & mask) == mask] |
| **5** | < (LT) [object < mask] |
| **6** | >= (GE) [object >= mask] |

**5.** Logical link between filters in a filter block

| | |
|---|---|
| **0** | OR |
| **1** | AND |

**6.** Logical link between filter blocks

| | |
|---|---|
| **0** | OR |
| **1** | AND |

**7.** Filter specification and validity

| | |
|---|---|
| **0** | Take all |
| **1** | trigger |
| **2** | pattern and offset |
| 4 | type |
| **8** | subtype |
| **12** | subtype & type *) |
| **16** | procid |
| **32** | contr & subcrate *) |
| **48** | contr & subcrate & procid *) |
| *) | byte/word sequ. from left to right |
| **0 - 12** | valid for events |
| **2 - 48** | valid for subevents |

**8.** Mask (=bit pattern)
decimal or hex (0x.....) enter here the
Word or LongWord (see 9.) req. value for 7.

**9.** Offset
decimal or hex (but like 8.)
enter here the required
value for 7.(2)
otherwise 0 (will be set automatically)

Definition (see also 8.):
LW: offset >= 0 index on event or subevent
(0: 1st LW, ..., etc.)
W: offset < 0 index on event or subevent
(-1: 2nd W, ..., etc.)

## Filter_examples

### Filter Examples for interactive or file input

```
! and /* are allowed comment declarations!


1.a) Output: whole event.
     Filter: Take all events with trigger >= 3
!
! output selection
1 ! block with 1 filter
1 0 1 0 0 0 0 0 0
! filter selection
1 ! block with 1 filter
1 1 0 6 0 0 1 3 0



1.b) is identical with 1.a)
!
1 ! block with 1 filter
1 1 1 6 0 0 1 3 0



2.a) Output: whole event.
     Filter: Take all events with
             trigger = 3 OR
             trigger = 7
       AND
             the first three bits set
                in the 15th LongWord of
```

```
                    subevent (processor id=20)
1 ! output selection
1 0 1 0 0 0 0 0 0
2 ! event filter selection
1 1 0 1 0 0 1 3 0
1 1 0 1 0 0 1 7 0
2 ! subevent filter selection
0 1 0 1 1 1 16 20 0
0 1 0 4 1 1 2 7 15 ! mask and offset decimal
```

```
2.b) Filter like 2.a) but
       Output subevent (processor id=20)
          and subevent (processor id=30)
2
0 0 1 1 0 0 16 10 0
0 0 1 1 0 0 16 20 0
2 ! event filter selection
1 1 0 1 0 0 1 3 0
1 1 0 1 0 0 1 7 0
2 ! subevent filter selection
0 1 0 1 1 1 16 20 0
0 1 0 4 1 1 2 0x7 0xF ! mask and offset hexadec.
```

## User_routines

These routines have to be provided and linked
together

## UANAL

| | |
|---|---|
| **Module** | UANAL.FOR |
| **CALLING** | UANAL(I4EVT, I2STS, I4LEN) |
| **PURPOSE** | User analysis routine. Here, histogramming etc. has to be done. |
| **PARAMETERS** | |
| **I4EVT** | INTEGER*4 I4EVT(0:I4LEN) event vector (Longwords) |

| | |
|---|---|
| **I4LEN** | INTEGER*4 I4LEN data length in LW |
| **I2STS** | INTEGER*2 I2STS Return status (0: success) |

## UCLINFO

| | |
|---|---|
| **Module** | UCLINFO.FOR |
| **CALLING** | UCLINFO(I4BUF1, I4BUF2) |
| **PURPOSE** | User info routine. Will be executed after the first and the last data buffer sent from the server and for each buffer with a CTRL_g keyboard input.<br><br>For details see UCLINFO.FOR template and GOOCINC:S_CLNTBUF.H for available info and statistics data. |

**PARAMETERS**

| | |
|---|---|
| **I4BUF1** | INTEGER*4 I4BUF1(0:*) and |
| **I4BUF2** | INTEGER*4 I4BUF2(0:*) ptrs to server and client info data |

## Implementation

| | |
|---|---|
| **PROCEDURES** | see PC_PROC |
| **STRUCTURES** | see PC_PROC Structures |
| **MACROS** | see PC_PROC Macros |
| **Return type** | none |
| **File name** | PC_CLIPAW.C |
| **Version** | 1.01 |
| **Author** | R.S. Mayer |
| **Last Update** | 03-May-1993 |

## Internals

| | |
|---|---|
| **Utility** | |
| **Module name** | F_CLIPAW |
| **File name** | PC_CLIPAW.C |
| **Home direct.** | TOOL$SOURCE |
| **Compile lib.** | TOOL$LIB:PC_GPS.TLB |
| **Link option** | SYS$LIBRARY:UCX$IPC.OLB/LIB SYS$SHARE:VAXCRTL.EXE/SHARE |
| **Created** | 01-Sep-1993 |

## Updates

| **Updates** | Date | Purpose |
|---|---|---|
| | **15-Dec-1993** | Input, output buffer structure changed<br>s_clntbuf_swap obsolet (RSM) |
| | **10-Jan-1994** | Documentation |
| | **02-Feb-1994** | Include names and defines modified!!! |
| | **03-Feb-1994** | New STC-routines |
| | **17-Feb-1994** | Handling read error (RSM) |
| | **25-Feb-1994** | UCLINFO reinserted! (RSM) |
| | **15-Mar-1994** | Server endian, adapted to new stc, RSM |
| | **11-Apr-1994** | struct tcpcomm no more extern.<br>Timeout bug removed. New function:<br>f_send_ackn. Sending ackn. buff rem.<br>from function f_read_server.<br>Ctrl_a,_c,_z now like last buffer.<br>Ackn. only after full treatment /RSM<br>: Some slight modifications /RSM |

### 5.0.1 PAW environment

On TOOL$SOURCE (VAX) you may find the following files:

- PAWMAIN.FOR
- UACTKD.FOR
- UANAL.FOR
- UCLINFO.FOR
- UKDEF.CDF

Then proceed as follows:

- To get the command definition: $ KUIPC UKDEF.CDF UKDEF.FOR
- Edit your own analysis program from UANAL.FOR. (You(!) must care about the byte order of the platform where UANAL shall work)
- Compile all FORTRAN files: $ for PAWMAIN,UACTKD,UANAL,UKDEF,UCLINFO
- Compile: $ cc tool$source:pc_clipaw.c+tool$lib:pc_gps.tlb/lib
- Compile: $ cc tool$source:pc_proc.c+tool$lib:pc_gps.tlb/lib
- link all with the option: sys$library:ucx$ipc.olb/lib, sys$share:vaxcrtl.exe/share. On VMS for the time being

**Remark:** On Alpha-VMS compile the C-sources with /STANDARD=VAXC. The link option is not required.

For further questions how to install your PAW environment, please contact the Data Analysis Group at GSI.

# Chapter 6

# GPS Client (standalone)

# PC_CLIENT

**PC_CLIENT(argc,argv)**

**PURPOSE** Standalone client for GOOSY - PAW - Server or SBS - Event - Server.
(See HELP MGOOPS and documentation m_event_serv .
For test purpose. For application, PAW etc. see F_CLIPAW

**ARGUMENTS**

| | |
|---|---|
| **argc** | Number of arguments + 1 |
| **argv[ 1]** | ptr to Node_name |
| **argv[ 2]** | ptr to Port number (>0) |
| **or** | using the port server |
| **argv[ 1]** | ptr to Server_name or Node_name::Server_name |
| **argv[ 2]** | ptr to Port number <= 0 |
| **argv[ 3]** | ptr to Number of events |
| **argv[ 4]** | ptr to Filter option or File name |
| **argv[ 5]** | ptr to Reduction rate |
| **argv[ 6]** | ptr to Echo rate |
| **argv[ 7]** | ptr to Display option |
| **argv[ 8]** | ptr to Buffer flushing time |

## Description

**FUNCTION** Client for test purpose GOOSY - PAW - Server and SBS - Event - Server.
(for detailed information see F_CLIPAW)

## Implementation

| | |
|---|---|
| **PROCEDURES** | see PC_PROC |
| **STRUCTURES** | see PC_PROC Structures |
| **MACROS** | see PC_PROC Macros |
| **Return type** | none |
| **File name** | PC_CLIENT.C |
| **Version** | 1.01 |
| **Author** | R.S. Mayer |
| **Last Update** | 11-Apr-1994 |

## Internals

| | |
|---|---|
| **Utility** | |
| **File name** | PC_CLIENT.C |
| **Home direct.** | TOOL$SOURCE |
| **Compile lib.** | TOOL$LIB:PC_GPS.TLB |
| **Link option** | SYS$LIBRARY:UCX$IPC.OLB/LIB (on VMS only) SYS$SHARE:VAXCRTL.EXE/SH (on VMS only) |
| **Created** | 01-Sep-1993 |

## Updates

| **Updates** | Date | Purpose |
|---|---|---|
| | **15-Dec-1993** | Input, output buffer structure changed s_clntbuf_swap obsolet (RSM) |
| | **20-Jan-1994** | _AIX compiles wo error (RSM) |
| | **21-Jan-1994** | modification first buffer handl. (RSM) |
| | **24-Jan-1994** | modifications (RSM) |
| | **02-Feb-1994** | Include names and defines modified!!! |
| | **03-Feb-1994** | New STC-routines |
| | **17-Feb-1994** | Handling read error (RSM) |

| | |
|---|---|
| **25-Feb-1994** | UCLINFO reinserted! (RSM) |
| **15-Mar-1994** | Server endian, adapted to new stc, RSM |
| **11-Apr-1994** | struct tcpcomm no more extern.<br>Timeout bug removed. New function:<br>f_send_ackn. Sending ackn. buff rem.<br>from function f_read_server.<br>Ctrl_a,_c,_z now like last buffer.<br>Ackn. only after full treatment /RSM<br>: Some slight modifications /RSM |

# Contents