TOW2023 - Seminar

FS Station Code

Alexander Neidhardt (TUM Wettzell)

Experience level: Advanced.

Description: This course describes how to write station specific code in C++. We discuss what is required to implement your own control loops outside of the FS supporting tasks required by the FS.

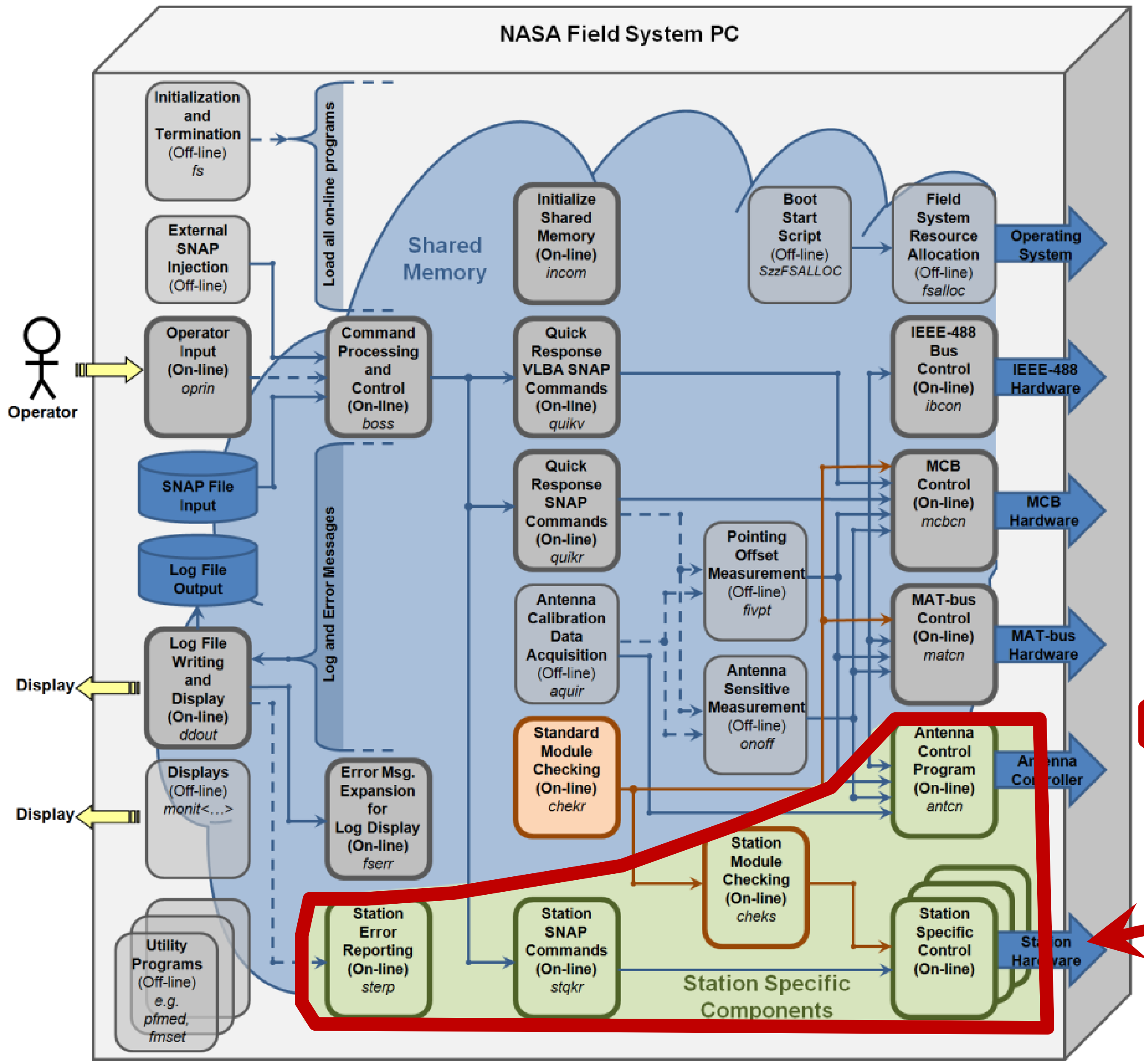
Code: FSa1, FSa2

TOW2023 - Seminar

FS Station Code

What does a station has to offer to the FS?
How to simplify access to data of the FS?
How to control your equipment from FS with “fsmonitor”?
How to command the FS in your code?
How to read answers from the FS in your code?
How to manually interact with the FS from remote?
How to use multiple devices in your FS?
How to integrate FS in your certified control loop?

What does a station has to offer to the FS?


NASA Field System PC
Initialization and Termination (Off-line) fs
External SNAP Injection (Off-line)
Operator Input (On-line) oprin
Operator
SNAP File Input
Log File Output
Log File Writing and Display (On-line) ddout
Display
Displays (Off-line) monit<...>
Display
Utility Programs (Off-line) e.g. pfmed, fmset
Load all on-line programs
Log and Error Messages
Shared Memory
Command Processing and Control (On-line) boss
Error Msg. Expansion for Log Display (On-line) fserr
Station Error Reporting (On-line) sterp
Initialize Shared Memory (On-line) incom
Quick Response VLBA SNAP Commands (On-line) quikv
Quick Response SNAP Commands (On-line) quikr
Antenna Calibration Data Acquisition (Off-line) aquir
Standard Module Checking (On-line) chekr
Station SNAP Commands (On-line) stqkr
Boot Start Script (Off-line) SzzFSALLOC
Pointing Offset Measurement (Off-line) fivpt
Antenna Sensitive Measurement (Off-line) onoff
Station Module Checking (On-line) cheks
Field System Resource Allocation (Off-line) fsalloc
IEEE-488 Bus Control (On-line) ibcon
MCB Control (On-line) mcbcn
MAT-bus Control (On-line) matcn
Antenna Control Program (On-line) antcn
Station Specific Control (On-line)
Station Specific Components
Operating System
IEEE-488 Hardware
MCB Hardware
MAT-bus Hardware
Antenna Controller
Station Hardware

The NASA Field System can be split into six main layers:

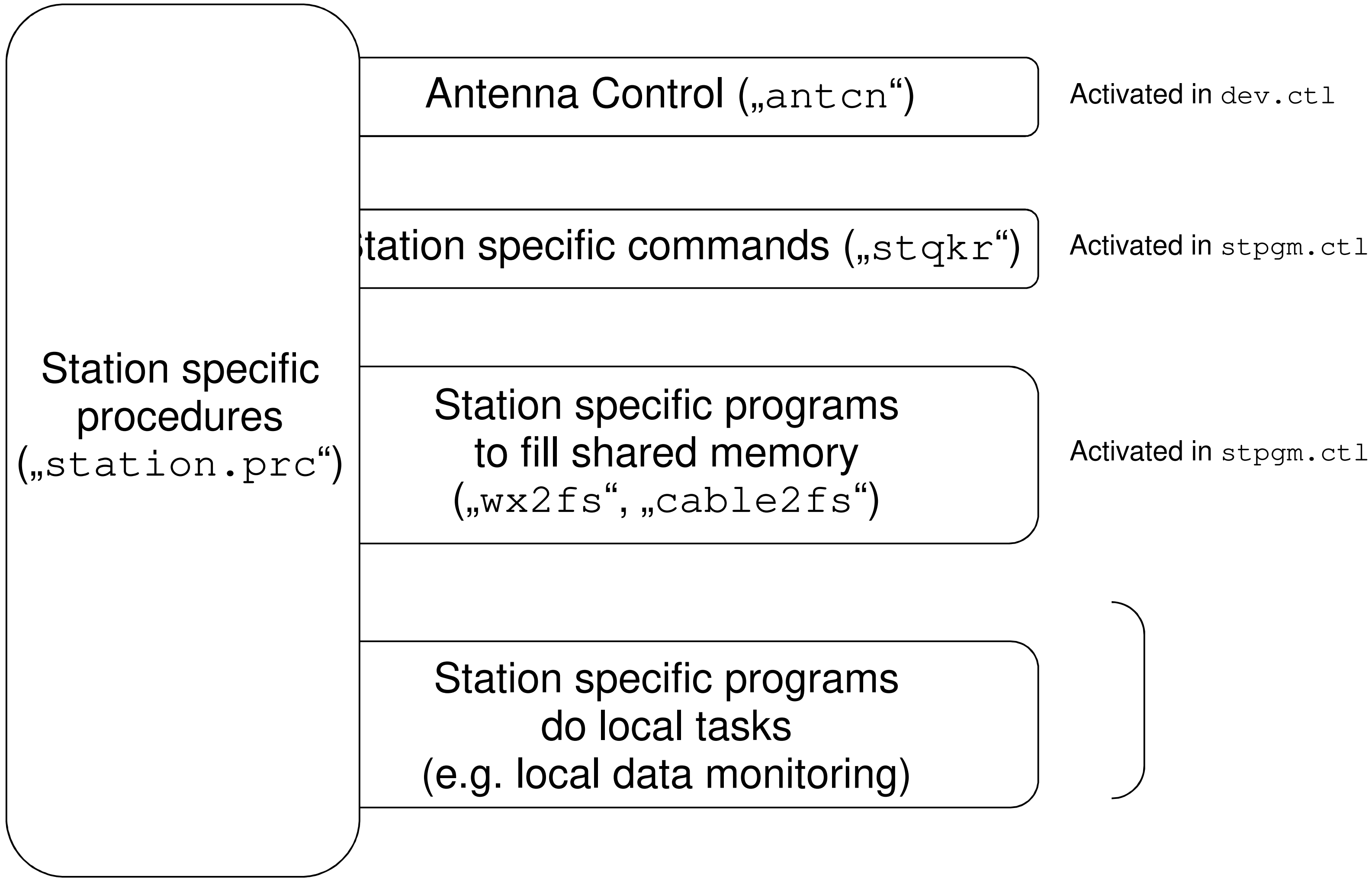
1. Programs for hardware control (hardware driving)
2. Programs for (module) checking (monitoring)
3. Programs for the SNAP command interpretation
4. Programs for Command Processing and Control (coordination: «boss» or, e.g., the Antenna Calibration Data Acquisition «aquir»)
5. Programs for error reporting
6. Programs for user interfacing

The NASA Field System can be split into two categories, according to where the code is developed:

1. the general Field System programs from NASA/NVI (Himwich, Horsley, et. al.
2. station code, individually programmed by station staff

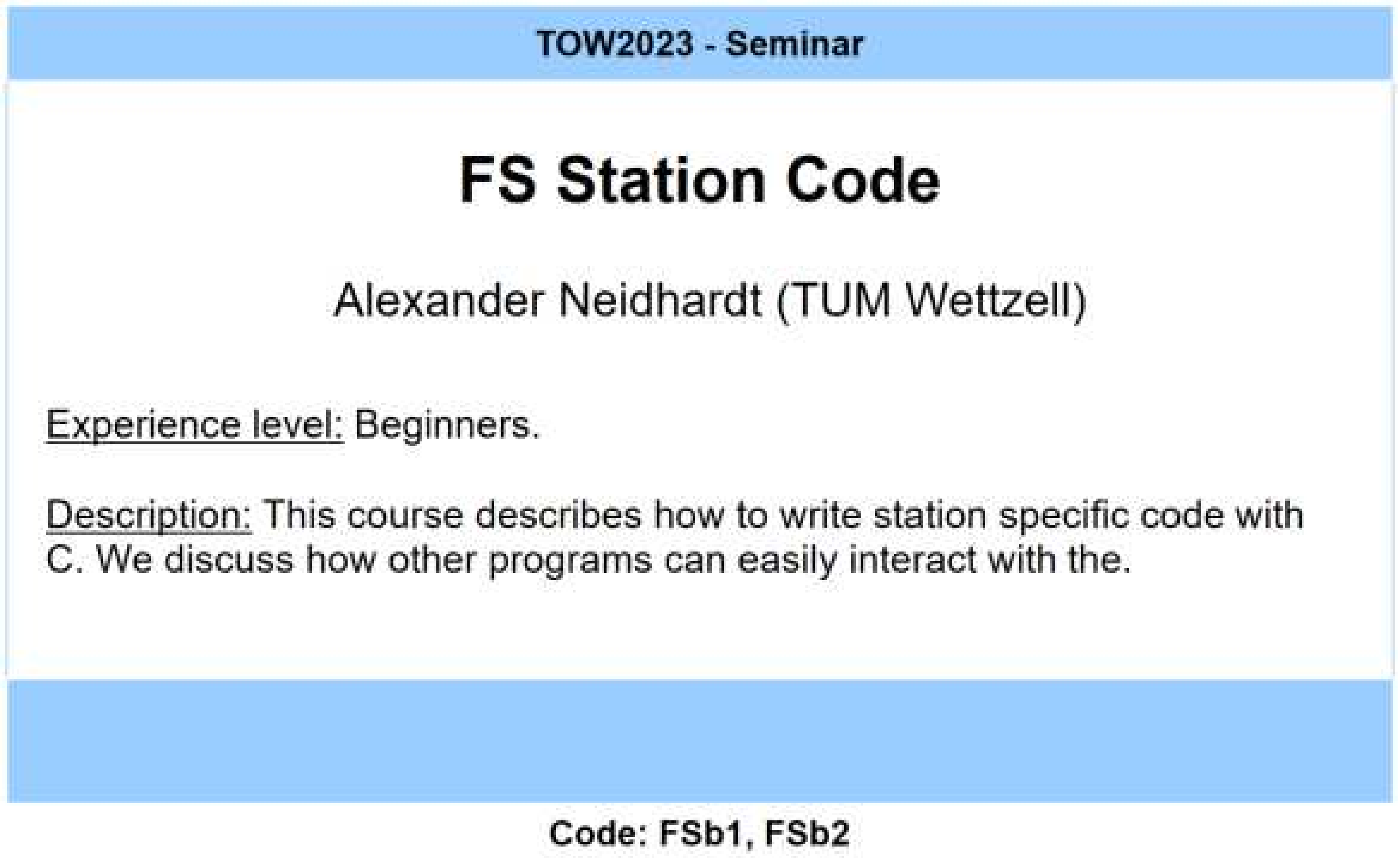
What does a station has to offer to the FS?

Station-specific programs


Station specific procedures („station.prc“)
Antenna Control („antcn“)
Activated in dev.ctl
tation specific commands („stqkr“)
Activated in stpgm.ctl
Station specific programs to fill shared memory („wx2fs“, „cable2fs“)
Activated in stpgm.ctl
Station specific programs do local tasks (e.g. local data monitoring)

What does a station has to offer to the FS?

For basics in FS programming see:



TOW2023 - Seminar

FS Station Code

Alexander Neidhardt (TUM Wettzell)

Experience level: Beginners.

Description: This course describes how to write station specific code with C. We discuss how other programs can easily interact with the.

Code: FSb1, FSb2

TOW2023 - Seminar

FS Station Code

What does a station has to offer to the FS?
How to simplify access to data of the FS?
How to control your equipment from FS with “fsmonitor”?
How to command the FS in your code?
How to read answers from the FS in your code?
How to manually interact with the FS from remote?
How to use multiple devices in your FS?
How to integrate FS in your certified control loop?

How to simplify access to data of the FS?

Module „fsmonitor“ (Wettzell)

Abstraction layer for NASA FS functionality
Based on code from Helge Rottmann
Written in C (=> usable for C and C++)
Avoiding some limitations, e.g. naming conventions, length of lines in the log

Consisting of:

- fsshm.c/.h, shmaccess_structs.h: Communicating with the Field System
    - Service functions (e.g. usInitSHM, usIsFieldSystemRunning)
    - Accessing shared memory values (e.g. usGetSHMTempWX, usSetSHMTempWX)
    - Communication with NASA FS boss (main program)
    - Command injection

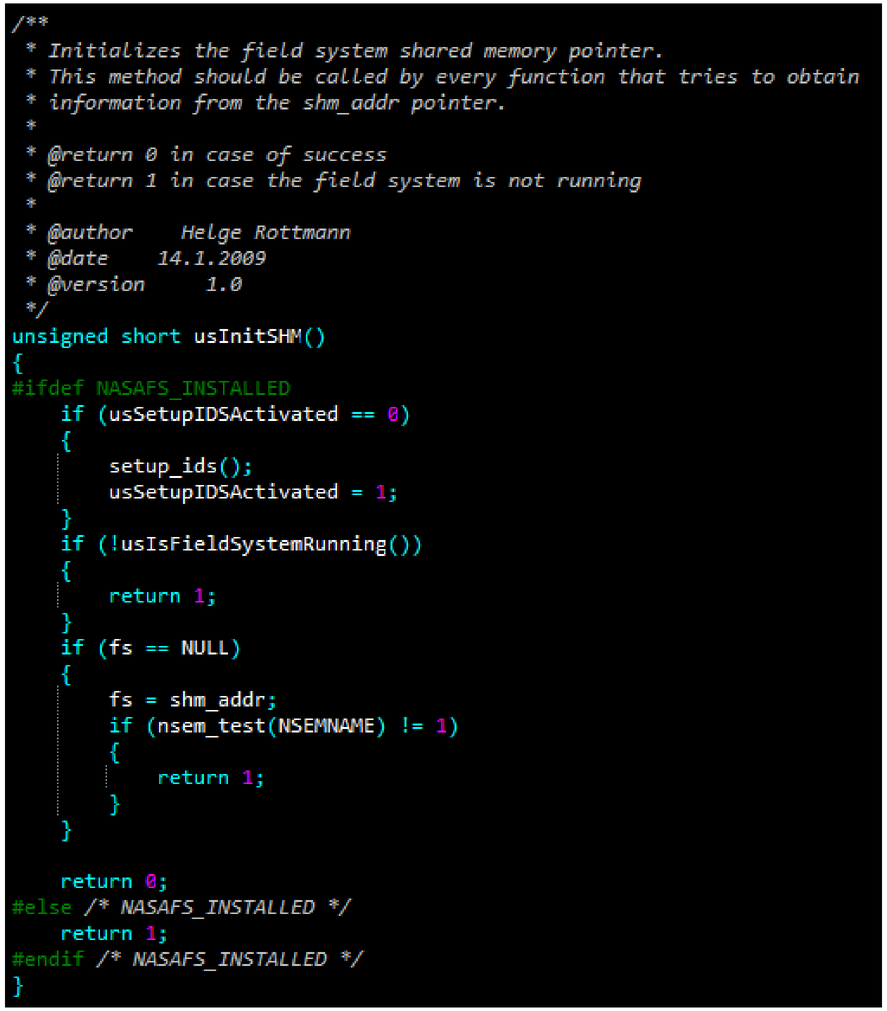
- fs_command.cpp/.hpp: Injection of commands to the Field System

- fs_util.cpp/.hpp: helping methods like converter etc.

How to simplify access to data of the FS?

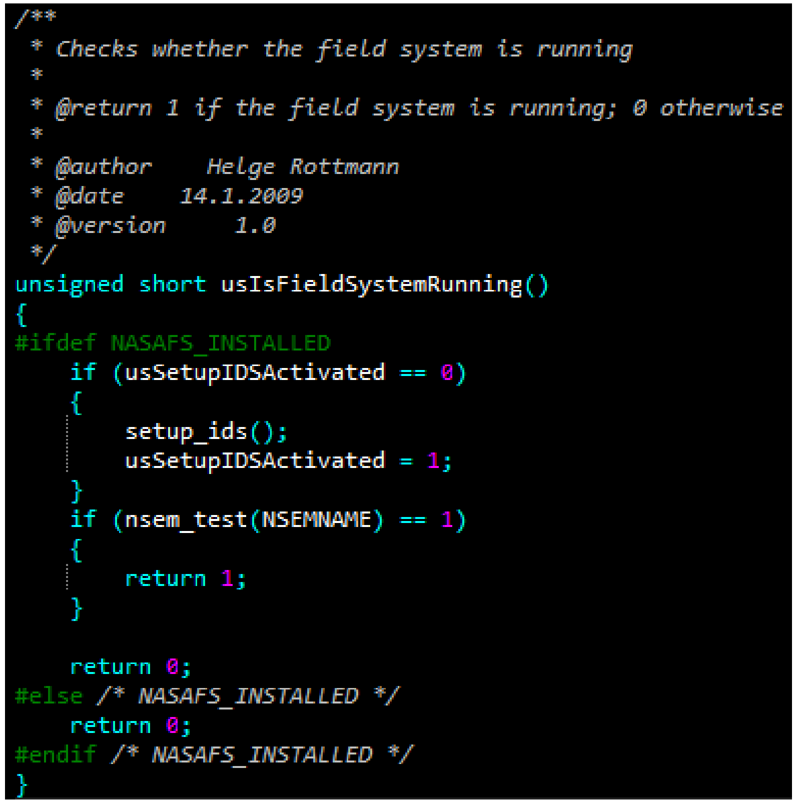
Module „fsmonitor“ (Wettzell)

Service functions



```
/**
 * Initializes the field system shared memory pointer.
 * This method should be called by every function that tries to obtain
 * information from the shm_addr pointer.
 *
 * @return 0 in case of success
 * @return 1 in case the field system is not running
 *
 * @author    Helge Rottmann
 * @date    14.1.2009
 * @version     1.0
 */
unsigned short usInitSHM()
{
#ifdef NASAFS_INSTALLED
    if (usSetupIDSActivated == 0)
    {
        setup_ids();
        usSetupIDSActivated = 1;
    }
    if (!usIsFieldSystemRunning())
    {
        return 1;
    }
    if (fs == NULL)
    {
        fs = shm_addr;
        if (nsem_test(NSEMNAME) != 1)
        {
            return 1;
        }
    }

    return 0;
#else /* NASAFS_INSTALLED */
    return 1;
#endif /* NASAFS_INSTALLED */
}
```



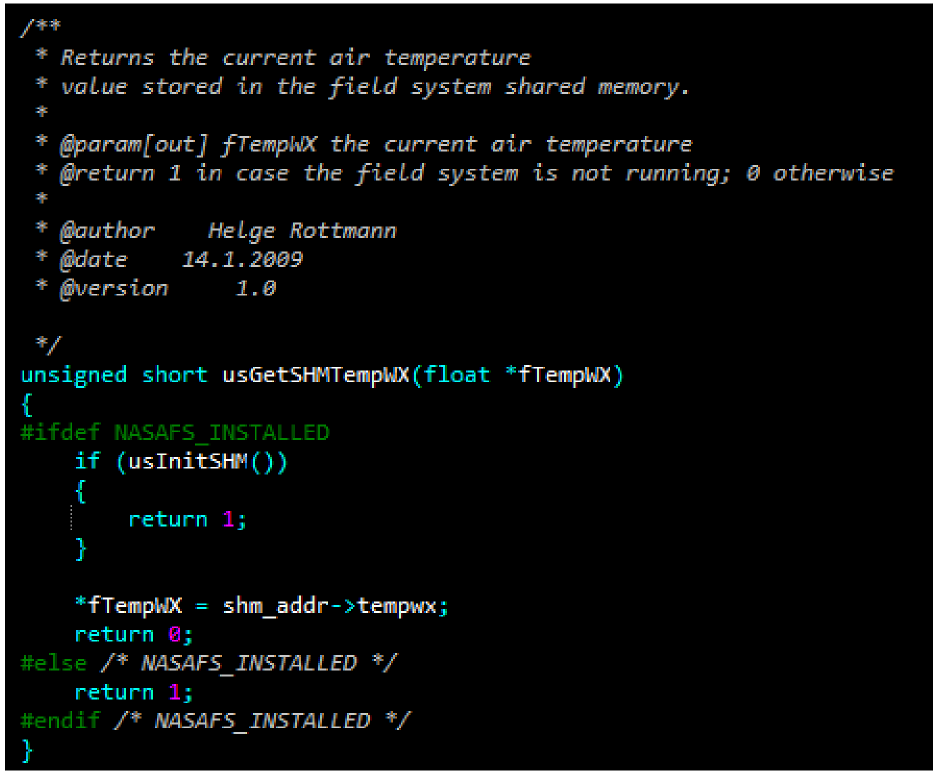
```
/**
 * Checks whether the field system is running
 *
 * @return 1 if the field system is running; 0 otherwise
 *
 * @author    Helge Rottmann
 * @date    14.1.2009
 * @version     1.0
 */
unsigned short usIsFieldSystemRunning()
{
#ifdef NASAFS_INSTALLED
    if (usSetupIDSActivated == 0)
    {
        setup_ids();
        usSetupIDSActivated = 1;
    }
    if (nsem_test(NSEMNAME) == 1)
    {
        return 1;
    }

    return 0;
#else /* NASAFS_INSTALLED */
    return 0;
#endif /* NASAFS_INSTALLED */
}
```

How to simplify access to data of the FS?

Module „fsmonitor“ (Wettzell)

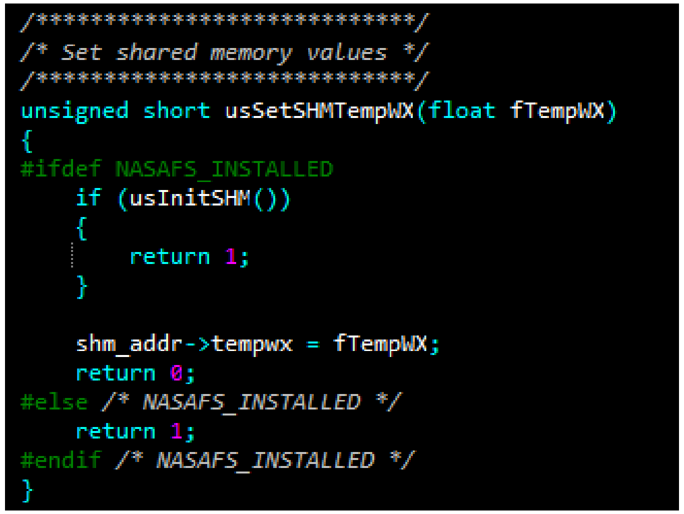
Accessing shared memory values



```
/**
 * Returns the current air temperature
 * value stored in the field system shared memory.
 *
 * @param[out] fTempWX the current air temperature
 * @return 1 in case the field system is not running; 0 otherwise
 *
 * @author    Helge Rottmann
 * @date    14.1.2009
 * @version     1.0

 */
unsigned short usGetSHMTempWX(float *fTempWX)
{
#ifdef NASAFS_INSTALLED
    if (usInitSHM())
    {
        return 1;
    }

    *fTempWX = shm_addr->tempwx;
    return 0;
#else /* NASAFS_INSTALLED */
    return 1;
#endif /* NASAFS_INSTALLED */
}
```



```
/******************************/
/* Set shared memory values */
/******************************/
unsigned short usSetSHMTempWX(float fTempWX)
{
#ifdef NASAFS_INSTALLED
    if (usInitSHM())
    {
        return 1;
    }

    shm_addr->tempwx = fTempWX;
    return 0;
#else /* NASAFS_INSTALLED */
    return 1;
#endif /* NASAFS_INSTALLED */
}
```

How to simplify access to data of the FS?

Module „fsmonitor“ (Wettzell)

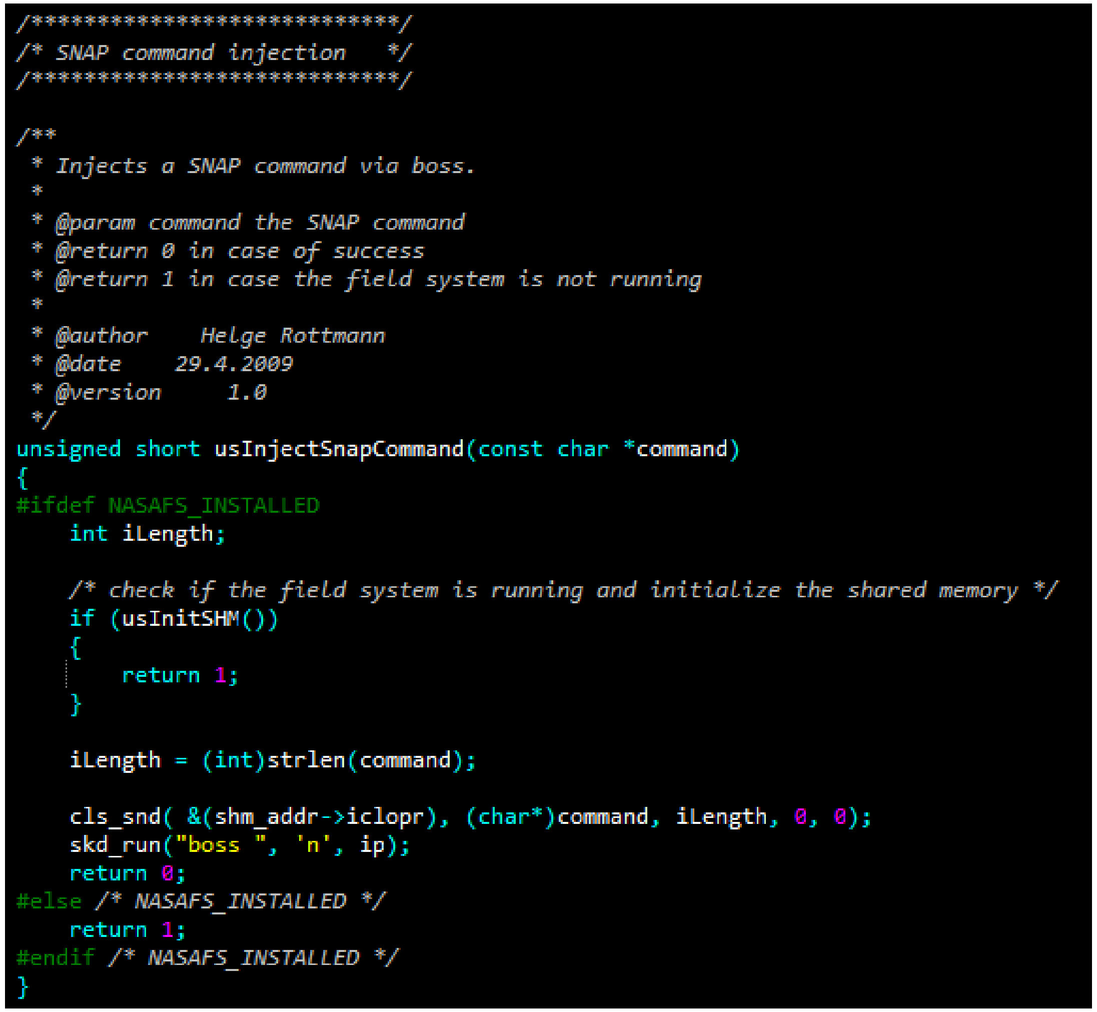
Communication with NASA FS boss

```
typedef long * BOSSCOM;
unsigned short usOpenBossCommunication (BOSSCOM * pBOSSCOMIdentifier);
unsigned short usCloseBossCommunication (BOSSCOM * pBOSSCOMIdentifier);
unsigned short usWaitForMessageFromBoss (BOSSCOM * pBOSSCOMIdentifier,
                                         char * pcProgramName,
                                         char acReceivedMessage[4096]);
long lGetCommandIdentifierOfIncomingCommand (BOSSCOM * pBOSSCOMIdentifier);
long lGetIPCClassNumberForIncomingMessage (BOSSCOM * pBOSSCOMIdentifier);
long lGetNumberOfElementsInIncomingMessage (BOSSCOM * pBOSSCOMIdentifier);
unsigned short usAcknowledgeMessageProcessing (BOSSCOM * pBOSSCOMIdentifier);
unsigned short usPrintMessage2Log (BOSSCOM * pBOSSCOMIdentifier,
                                   const char acProgramNameInput[6],
                                   const char * pcFormat,
                                   ...);
unsigned short usPrintError2Log (BOSSCOM * pBOSSCOMIdentifier,
                                 const char acProgramNameInput[6],
                                 const char acFSErrorIdentCodeInput[3],
                                 int iErrorCode,
                                 const char * pcFormat,
                                 ...);
unsigned short usReplyMessageToBoss (BOSSCOM * pBOSSCOMIdentifier,
                                     const char * pcFormat,
                                     ...);
unsigned short usWriteSatEphemToFile(void);
```

How to simplify access to data of the FS?

Module „fsmonitor“ (Wettzell)

Command injection



```
/*****************************/
/* SNAP command injection    */
/*****************************/

/**
 * Injects a SNAP command via boss.
 *
 * @param command the SNAP command
 * @return 0 in case of success
 * @return 1 in case the field system is not running
 *
 * @author    Helge Rottmann
 * @date    29.4.2009
 * @version     1.0
 */
unsigned short usInjectSnapCommand(const char *command)
{
#ifdef NASAFS_INSTALLED
    int iLength;

    /* check if the field system is running and initialize the shared memory */
    if (usInitSHM())
    {
        return 1;
    }

    iLength = (int)strlen(command);

    cls_snd( &(shm_addr->iclopr), (char*)command, iLength, 0, 0);
    skd_run("boss ", 'n', ip);
    return 0;
#else /* NASAFS_INSTALLED */
    return 1;
#endif /* NASAFS_INSTALLED */
}
```

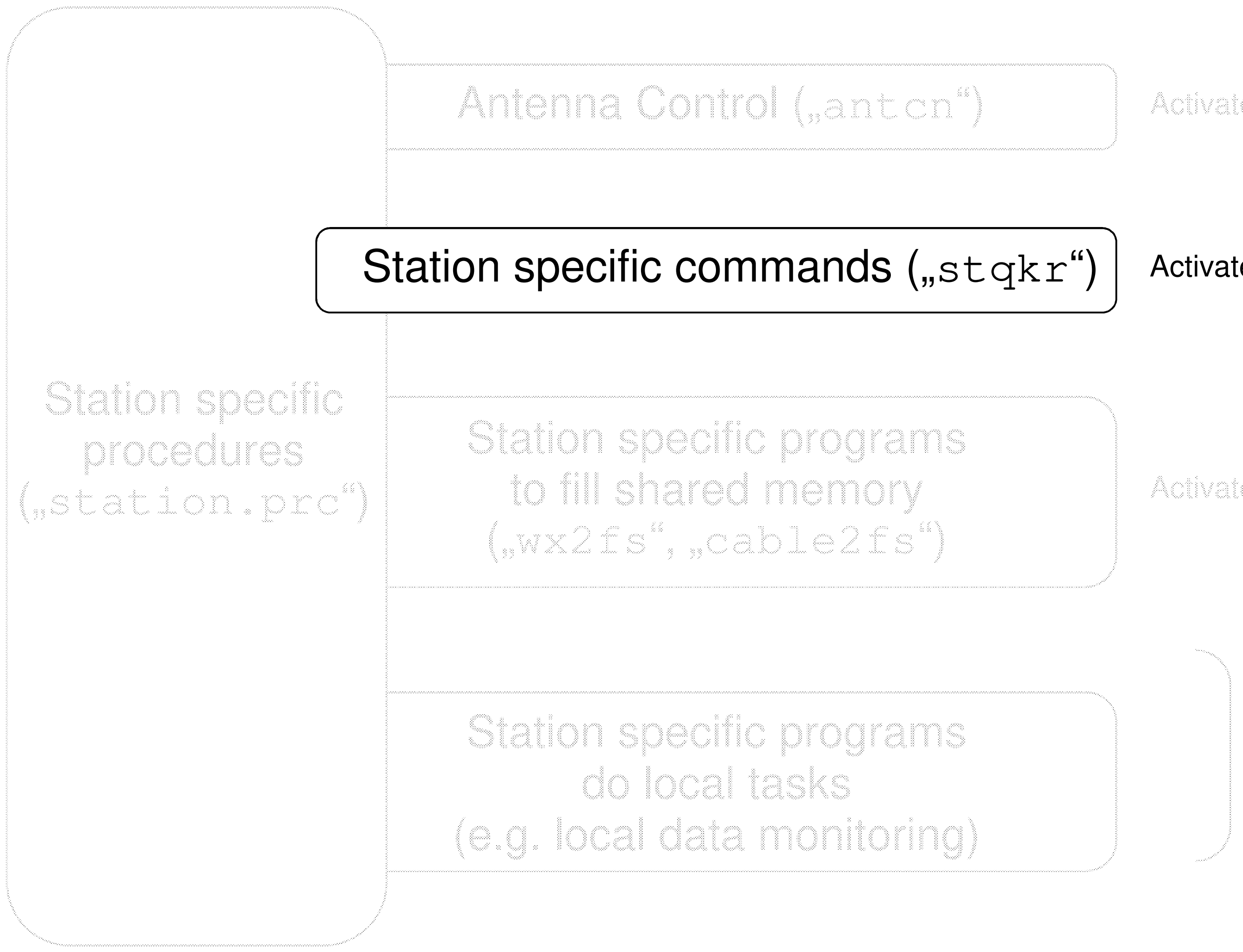
TOW2023 - Seminar

FS Station Code

What does a station has to offer to the FS?
How to simplify access to data of the FS?
How to control your equipment from FS with “fsmonitor”?
How to command the FS in your code?
How to read answers from the FS in your code?
How to manually interact with the FS from remote?
How to use multiple devices in your FS?
How to integrate FS in your certified control loop?

How to control your equipment from FS with “fsmonitor”?

Station-specific programs


Station specific procedures („station.prc“)
Antenna Control („antcn“)
Activated in dev.ctl
Station specific commands („stqkr“)
Activated in stpgm.ctl
Station specific programs to fill shared memory („wx2fs“, „cable2fs“)
Activated in stpgm.ctl
Station specific programs do local tasks (e.g. local data monitoring)

How to control your equipment from FS with “fsmonitor”?

„stqkr.cpp“

```
#include <stdio.h>
#include <stdlib.h>
#include "fsshm.h„
#include "simple_structured_conf.hpp"
#include "meteo.hpp"
#include "rxmon.hpp"
#include "testequip.hpp"

int main (int iArgc, char * pcArgv[])
{
    BOSSCOM BOSSCOMIdentifier = NULL;  /// BOSSCOMIdentifier = Identification of current communication (it is a pointer to the data set
                                       /// "ip" of the field system)
    char acProgramName[6] = "stqkr";   /// pcProgramName = name of the program, which waits for a message
    char acReceivedMessage[4096];      /// acReceivedMessage = complete message sent from boss
    CSimpleStructuredConf CConfiguration; /// CConfiguration = the whole configuration parameters of the stqkr program
    unsigned long ulCurrentLineNumber; /// ulCurrentLineNumber = line number of the configuration file, where an error occured
    std::string strCurrentTag;         /// strCurrentTag = tag in the configuration file, where the error occured

    printf ("stqkr: Startup ...\n");

    /// Open communication to NASA FS boss
    if (usOpenBossCommunication (&BOSSCOMIdentifier))
    {
        printf ("[ERROR] stqkr: Cannot open connection to NASA FS boss\n");
        return 1;
    }

    /// Check program parameters
    if (iArgc != 2)
    {
        (void) usPrintError2Log (&BOSSCOMIdentifier, acProgramName, "SQ", -1, "");
        goto CloseBossCommunication;
    }

    /// Read configuration
    if (CConfiguration.usReadConfig (pcArgv[1], ulCurrentLineNumber, strCurrentTag))
    {
        if (strCurrentTag.empty())
        {
            (void) usPrintError2Log (&BOSSCOMIdentifier, acProgramName, "SQ", -2, "No file found");
        }
        else
        {
            (void) usPrintError2Log (&BOSSCOMIdentifier, acProgramName, "SQ", -2, "Error in line %ld arround tag '%s'",
                                     ulCurrentLineNumber, strCurrentTag.c_str());
        }
        goto CloseBossCommunication;
    }
```

Initialize

How to control your equipment from FS with “fsmonitor”?

„stqkr.cpp“

```
    /// Start processing loop
    while (1)
    {
        /// Wait for incoming messages from boss
        if (usWaitForMessageFromBoss (&BOSSCOMIdentifier,
                                      acProgramName,
                                      acReceivedMessage))
        {
            continue;
        }

        /// Switch between the different commands according to the class number
        switch (lGetIPCClassNumberForIncomingMessage (&BOSSCOMIdentifier))
        {
            // *********************************************************
            // Command "wx"
            // *********************************************************
            case 100: /* wx */
            {
            }
            // *********************************************************
            // Command "rx" and "rxall"
            // *********************************************************
            case 200: /* rx */
            case 201: /* rxall */
            {
            }
            ...

        if (usAcknowledgeMessageProcessing (&BOSSCOMIdentifier))
        {
            continue;
        }
    }

CloseBossCommunication:
    /// Close communication to NASA FS boss
    if (usCloseBossCommunication (&BOSSCOMIdentifier))
    {
        printf ("[ERROR] stqkr: Cannot close connection\n");
        return 1;
    }

    printf ("[ERROR] stqkr: While loop failed\n");
    return 1;
}
```

Wait for incoming Orders and Receive command and arguments

Identify command

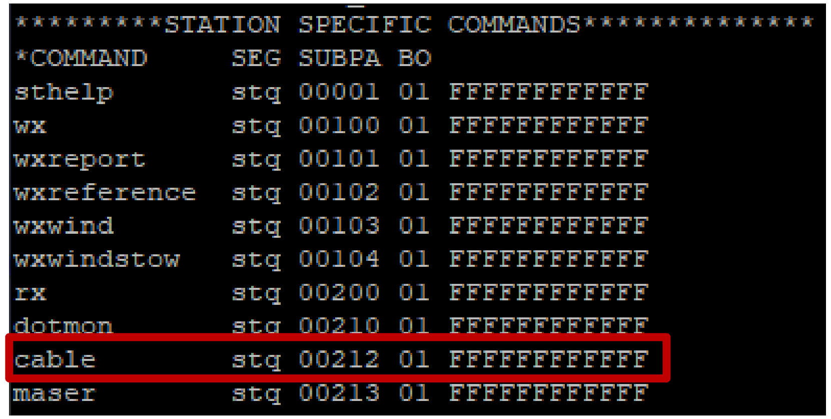
Interprete and Performe action according to command => Call function

Prepare return values

How to control your equipment from FS with “fsmonitor”?

„stqkr.cpp“ – sample SNAP command „dotmon“

/usr2/control/stcmd.ctl



```
*********STATION SPECIFIC COMMANDS**************
*COMMAND      SEG SUBPA BO
sthelp        stq 00001 01 FFFFFFFFFFFF
wx            stq 00100 01 FFFFFFFFFFFF
wxreport      stq 00101 01 FFFFFFFFFFFF
wxreference   stq 00102 01 FFFFFFFFFFFF
wxwind        stq 00103 01 FFFFFFFFFFFF
wxwindstow    stq 00104 01 FFFFFFFFFFFF
rx            stq 00200 01 FFFFFFFFFFFF
dotmon        stq 00210 01 FFFFFFFFFFFF
cable         stq 00212 01 FFFFFFFFFFFF
maser         stq 00213 01 FFFFFFFFFFFF
```

stqkr.cpp

```
switch (lGetIPCClassNumberForIncomingMessage (&BOSSCOMIdentifier))
{
...
            case 210:
            {
                /// Call function from testequip.cpp/.hpp
                if (usGetDotmon (&BOSSCOMIdentifier, acReceivedMessage))
                {
                    (void) usPrintError2Log (&BOSSCOMIdentifier, acProgramName, "SQ", -800, "");
                }
                break;
            }
...
}
```

How to control your equipment from FS with “fsmonitor”?

„stqkr.cpp“ – sample SNAP command „dotmon“

```
unsigned short usGetDotmon (BOSSCOM * pBOSSCOMIdentifier,
                            std::string strCommand)
{
    /// <b> Variables: </b>
    unsigned short usRetVal = 0;
    bool bConnectionOpened = false;
    SimpleSocketType SClientSocket;
    char acBuffer[1024];
    unsigned long ulBufferLength;
    double dTimeValueInSec = 0.0;

    /// <b> Operations: </b>
    /// Open connection
    /// Send command for reading the last actual counter value
    /// Receive counter value
    /// Send answer to NASA FS
    if (usReplyMessageToBoss (pBOSSCOMIdentifier, "gps-fmout/%.4e", dTimeValueInSec) ||
        usPrintMessage2Log (pBOSSCOMIdentifier, "stqkr", "Info: gps (Symmetricom SyncServer S250) minus fmout (Mark5B+)"))
    {
        (void) usPrintError2Log (pBOSSCOMIdentifier, "stqkr", "SQ", -800, "dotmon: Cannot return dotmon data");
        usRetVal = 1;
        goto ReturnToStqkr;
    }
    ...
    if (bConnectionOpened)
    {
        if(uiCloseSocket(&SClientSocket))
        {
            (void) usPrintError2Log (pBOSSCOMIdentifier, "stqkr", "SQ", -800, "dotmon: Cannot close socket to 192.168.208.8:5025");
            usRetVal = 1;
        }
        bConnectionOpened = false;
    }
    /// Send error answer to NASA FS
    if (usRetVal)
    {
        if (usReplyMessageToBoss (pBOSSCOMIdentifier, "dotmon: MK5B-dotmon to HOUSE-PPS/NOK"))
        {
            (void) usPrintError2Log (pBOSSCOMIdentifier, "stqkr", "SQ", -800, "dotmon: Cannot return dotmon data");
            usRetVal = 1;
        }
    }

    return usRetVal;
}
```

How to control your equipment from FS with “fsmonitor”?

„stqkr.cpp“ – sample SNAP command „dotmon“

```
unsigned short usGetDotmon (BOSSCOM * pBOSSCOMIdentifier,
                            std::string strCommand)
{
    /// <b> Variables: </b>
    unsigned short usRetVal = 0;
    bool bConnectionOpened = false;
    SimpleSocketType SClientSocket;
    char acBuffer[1024];
    unsigned long ulBufferLength;
    double dTimeValueInSec = 0.0;

    /// <b> Operations: </b>
    /// Open connection
    /// Send command for reading the last actual counter value
    /// Receive counter value
    /// Send answer to NASA FS
    if (usReplyMessageToBoss (pBOSSCOMIdentifier, "gps-fmout/%.4e", dTimeValueInSec) ||
        usPrintMessage2Log (pBOSSCOMIdentifier, "stqkr", "Info: gps (Symmetricom SyncServer S250) minus fmout (Mark5B+)"))
    {
        (void) usPrintError2Log (pBOSSCOMIdentifier, "stqkr", "SQ", -800, "dotmon: Cannot return dotmon data");
        usRetVal = 1;
        goto ReturnToStqkr;
    }
    ...
    if (bConnectionOpened)
    {
        if(uiCloseSocket(&SClientSocket))
        {
            (void) usPrintError2Log (pBOSSCOMIdentifier, "stqkr", "SQ", -800, "dotmon: Cannot close socket to 192.168.208.8:5025");
            usRetVal = 1;
        }
        bConnectionOpened = false;
    }
    /// Send error answer to NASA FS
    if (usRetVal)
    {
        if (usReplyMessageToBoss (pBOSSCOMIdentifier, "dotmon: MK5B-dotmon to HOUSE-PPS/NOK"))
        {
            (void) usPrintError2Log (pBOSSCOMIdentifier, "stqkr", "SQ", -800, "dotmon: Cannot return dotmon data");
            usRetVal = 1;
        }
    }

    return usRetVal;
}
```

Error processing comparable to „printf“

TOW2023 - Seminar

FS Station Code

What does a station has to offer to the FS?
How to simplify access to data of the FS?
How to control your equipment from FS with “fsmonitor”?
How to command the FS in your code?
How to read answers from the FS in your code?
How to manually interact with the FS from remote?
How to use multiple devices in your FS?
How to integrate FS in your certified control loop?

How to command the FS in your code?

Command injection

`/usr2/fs/bin/inject_snap`

```
/usr2/fs/bin/inject_snap -w log
```

➔ Output of current log filename
e.g. `log/station`

```
/usr2/fs/bin/inject_snap <command>
e.g.
/usr2/fs/bin/inject_snap wx
```

➔ Send command to FS

```
/usr2/fs/bin/inject_snap '" Test'
```

➔ Send comment to FS

How to command the FS in your code?

Command injection

`/usr2/fs/bin/inject_snap`

Sample code in Perl:

```
# ============================================================================================
# Subroutine: get_logfile_path
# Parameter:  $rLogfilePath <- name of logfile
# Parameter:  $rProtocol <- protocol of errors and work
# Return:     <- Error code (0=ok, 1=error)
# ============================================================================================
sub get_logfile_path {
    my ($rLogfilePath, $rProtocol)=@_;
    my @command = qw(/usr2/fs/bin/inject_snap -w log);
    my $stdin = "";
    my $stdout = "";
    my $stderr = "";

    # Init return value
    $$rLogfilePath = "";

    # Run command
    run3 (\@command, \$stdin, \$stdout, \$stderr);
    if ($?) {
        $$rProtocol = $$rProtocol."[ERROR] Reading of log-file name from NASA FS is not possible\n";
        return 1;
    }

    # Analyse return output
    if ($stdout =~ /log\/.*/) {
        ($$rLogfilePath) = ($stdout =~ /log\/([^\s]+)\s*/);
    $$rLogfilePath = "/usr2/log/".$$rLogfilePath.".log";
    }
    else {
        $$rProtocol = $$rProtocol."[ERROR] Returned log-file name from NASA FS is wrong\n";
        return 1;
    }

    $$rProtocol = $$rProtocol."Current log-file is ".$$rLogfilePath."\n";
    return 0;
}
```

`inject_snap` `–w` will not wait for a procedure to finish

TOW2023 - Seminar

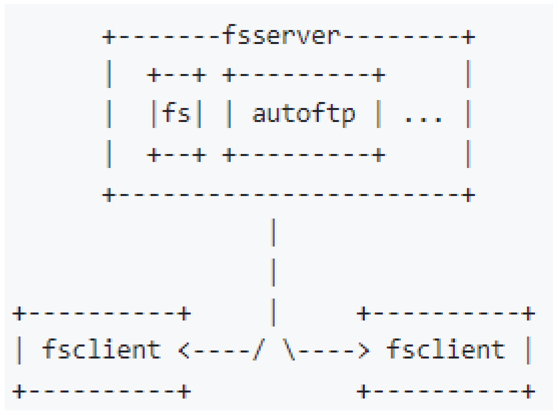
FS Station Code

What does a station has to offer to the FS?
How to simplify access to data of the FS?
How to control your equipment from FS with “fsmonitor”?
How to command the FS in your code?
How to read answers from the FS in your code?
How to manually interact with the FS from remote?
How to use multiple devices in your FS?
How to integrate FS in your certified control loop?

How to read answers from the FS in your code?

Command reply catching („streamlog“)

FS Display Server to get log messages



```
        +-------fsserver--------+
        |  +--+ +---------+     |
        |  |fs| | autoftp | ... |
        |  +--+ +---------+     |
        +-----------------------+
                   |
                   |
+----------+       |       +----------+
| fsclient <----/ \----> fsclient |
+----------+               +----------+
```

“streamlog” is recommended for FS 10.2 or greater
to get log messages.

https://github.com/nvi-inc/fs/blob/main/misc/display_server.md

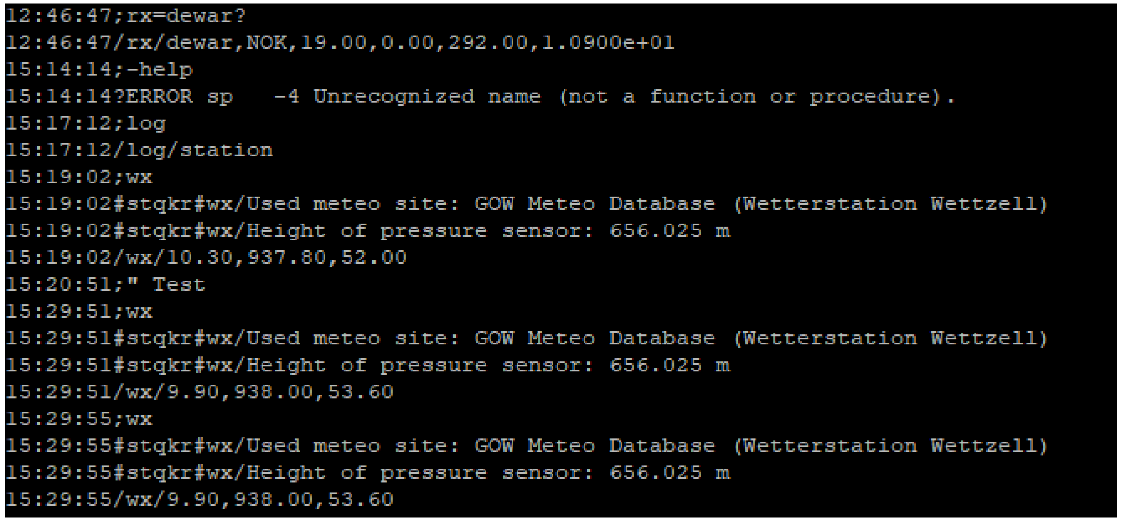
How to read answers from the FS in your code?

Command reply catching („streamlog“)

`/usr2/fs/bin/fsclient`

`/usr2/fs/bin/fsclient -n` ➔ Get all log messages on standard out

`/usr2/fs/bin/fsclient -s -n` ➔ Get all log messages on standard out inclusively some historic lines



```
12:46:47;rx=dewar?
12:46:47/rx/dewar,NOK,19.00,0.00,292.00,1.0900e+01
15:14:14;-help
15:14:14?ERROR sp   -4 Unrecognized name (not a function or procedure).
15:17:12;log
15:17:12/log/station
15:19:02;wx
15:19:02#stqkr#wx/Used meteo site: GOW Meteo Database (Wetterstation Wettzell)
15:19:02#stqkr#wx/Height of pressure sensor: 656.025 m
15:19:02/wx/10.30,937.80,52.00
15:20:51;" Test
15:29:51;wx
15:29:51#stqkr#wx/Used meteo site: GOW Meteo Database (Wetterstation Wettzell)
15:29:51#stqkr#wx/Height of pressure sensor: 656.025 m
15:29:51/wx/9.90,938.00,53.60
15:29:55;wx
15:29:55#stqkr#wx/Used meteo site: GOW Meteo Database (Wetterstation Wettzell)
15:29:55#stqkr#wx/Height of pressure sensor: 656.025 m
15:29:55/wx/9.90,938.00,53.60
```

==> pipe the output to a program which can read it as standard in so that you can use it in other programs or scripts e.g.

`fsclient -s -n | grep "wx"`

End with Ctrl - C

TOW2023 - Seminar

FS Station Code

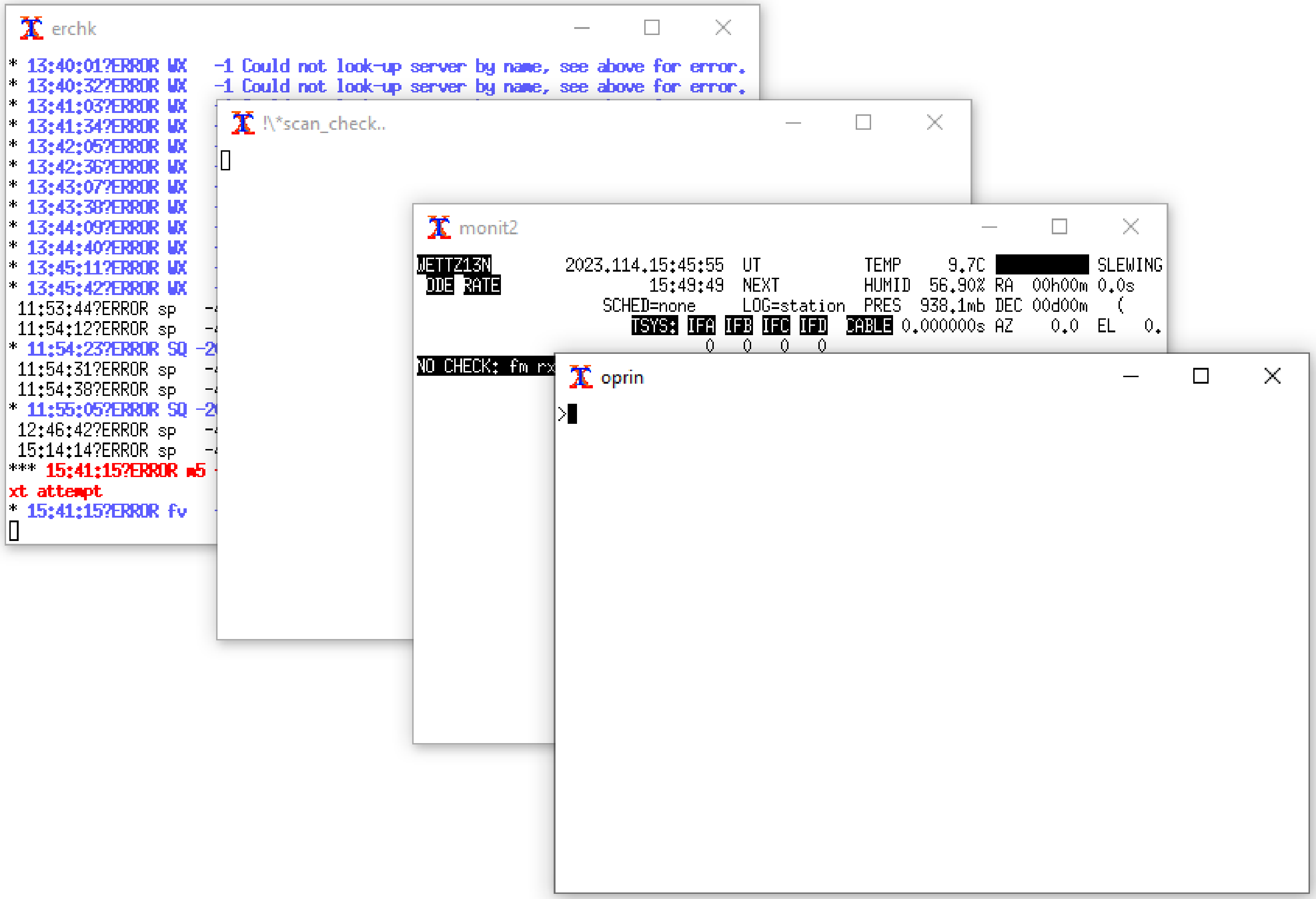
What does a station has to offer to the FS?
How to simplify access to data of the FS?
How to control your equipment from FS with “fsmonitor”?
How to command the FS in your code?
How to read answers from the FS in your code?
How to manually interact with the FS from remote?
How to use multiple devices in your FS?
How to integrate FS in your certified control loop?

How to manually interact with the FS from remote?

Standard remote control:

`/usr2/fs/bin/fsclient` via SSH-connection and X-forwarding

`fsclient` ➔ Start the client windows



```
erchk
* 13:40:01?ERROR WX   -1 Could not look-up server by name, see above for error.
* 13:40:32?ERROR WX   -1 Could not look-up server by name, see above for error.
* 13:41:03?ERROR WX   -
* 13:41:34?ERROR WX   -
* 13:42:05?ERROR WX   -
* 13:42:36?ERROR WX   -
* 13:43:07?ERROR WX   -
* 13:43:38?ERROR WX   -
* 13:44:09?ERROR WX   -
* 13:44:40?ERROR WX   -
* 13:45:11?ERROR WX   -
* 13:45:42?ERROR WX   -
  11:53:44?ERROR sp   -
  11:54:12?ERROR sp   -
* 11:54:23?ERROR SQ -2
  11:54:31?ERROR sp   -
  11:54:38?ERROR sp   -
* 11:55:05?ERROR SQ -2
  12:46:42?ERROR sp   -
  15:14:14?ERROR sp   -
*** 15:41:15?ERROR m5 -
xt attempt
* 15:41:15?ERROR fv   -
```

```
!\*scan_check..
```

```
monit2
WETTZ13N         2023.114.15:45:55  UT            TEMP     9.7C            SLEWING
 ODE RATE                 15:49:49  NEXT          HUMID  56.90% RA  00h00m 0.0s
                    SCHED=none      LOG=station   PRES  938.1mb DEC 00d00m   (
                       TSYS: IFA IFB IFC IFD  CABLE 0.000000s AZ     0.0  EL    0.
                               0   0   0   0
NO CHECK: fm rx
```

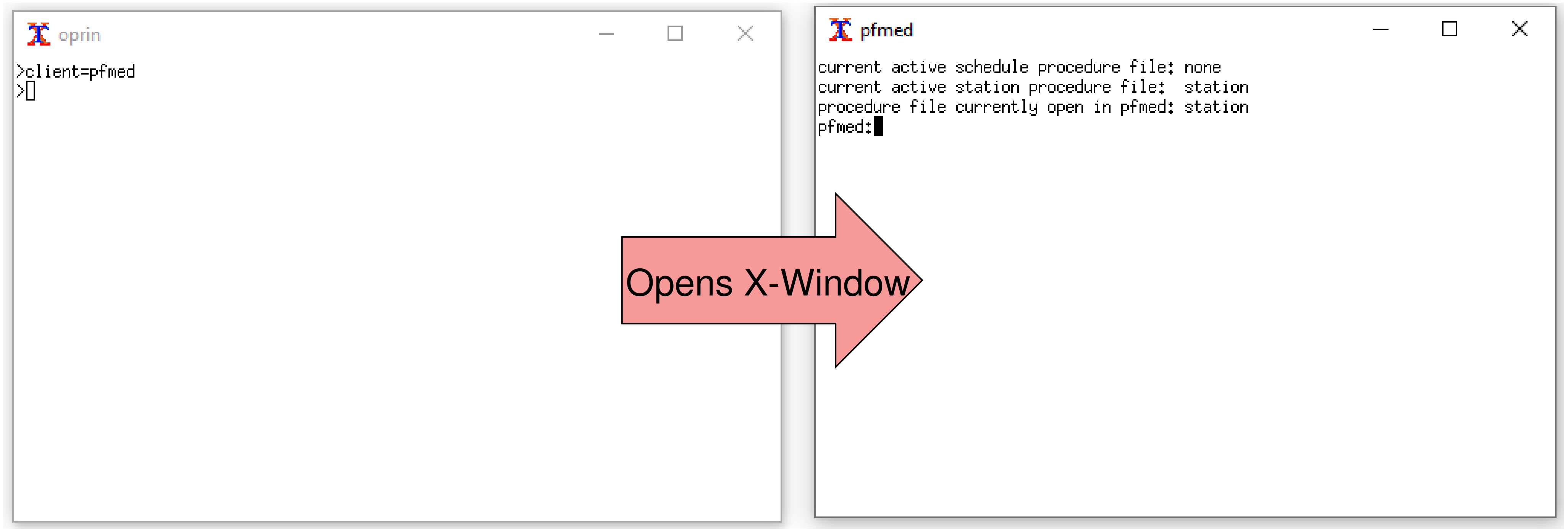
```
oprin
>
```

How to manually interact with the FS from remote?

Standard remote control:

`/usr2/fs/bin/fsclient` via SSH-connection and X-forwarding

`client=` commands in fsclient-oprin (not in „oprin“-call in a shell)



```
>client=pfmed
>
```

```
current active schedule procedure file: none
current active station procedure file:  station
procedure file currently open in pfmed: station
pfmed:
```

Control your FS from MS Windows:
- Install X-Window-Server, e.g. Xming or VcXsrv Windows X Server
- Connect to the FS PC using SSH with X-forwarding
- Start „fsclient“

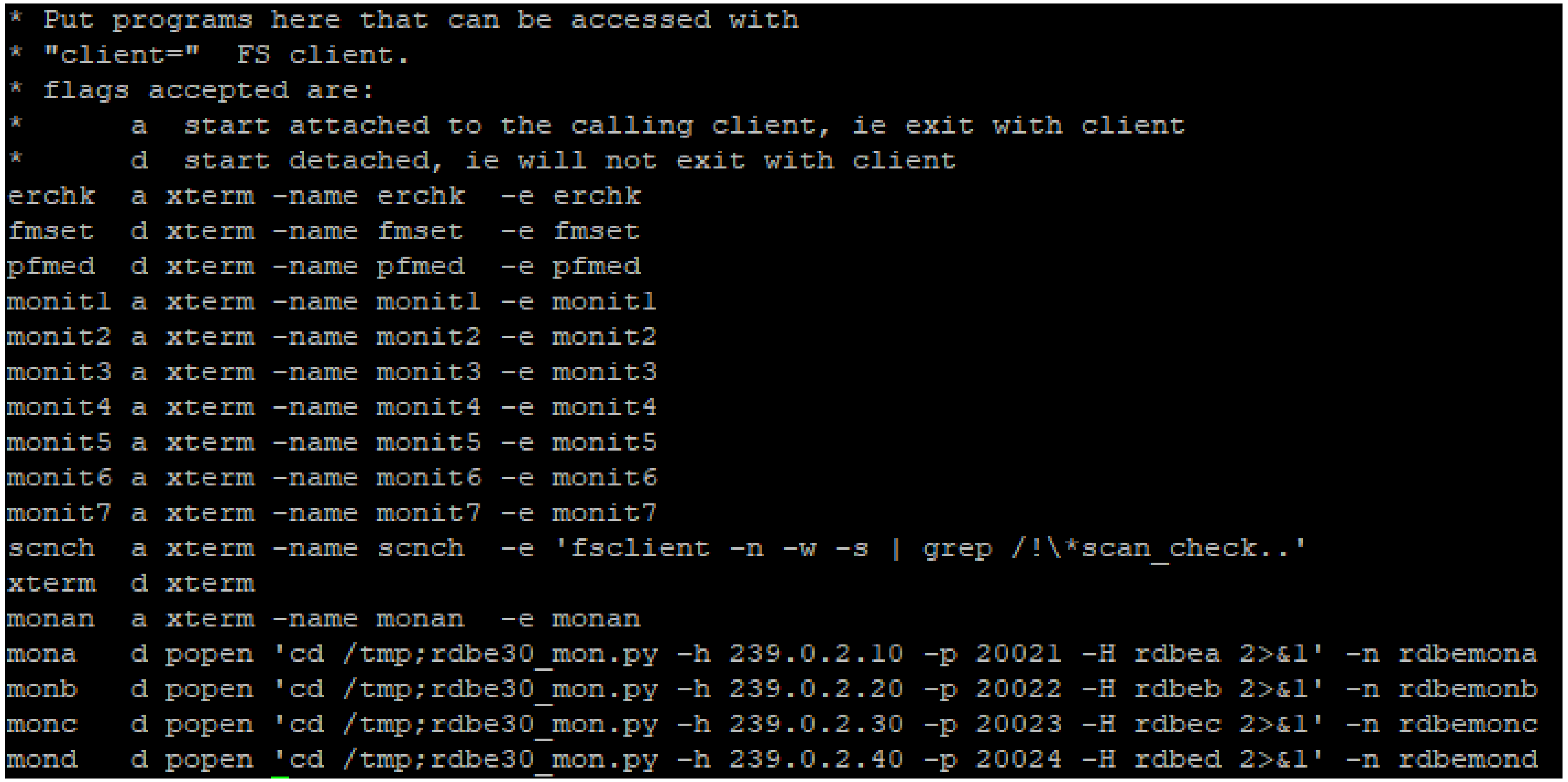
How to manually interact with the FS from remote?

Standard remote control:

`/usr2/fs/bin/fsclient` via SSH-connection and X-forwarding

`client=` commands in fsclient-oprin

`/usr2/control/clpgm.ctl`



```
* Put programs here that can be accessed with
* "client="  FS client.
* flags accepted are:
*       a  start attached to the calling client, ie exit with client
*       d  start detached, ie will not exit with client
erchk   a xterm -name erchk  -e erchk
fmset   d xterm -name fmset  -e fmset
pfmed   d xterm -name pfmed  -e pfmed
monit1  a xterm -name monit1 -e monit1
monit2  a xterm -name monit2 -e monit2
monit3  a xterm -name monit3 -e monit3
monit4  a xterm -name monit4 -e monit4
monit5  a xterm -name monit5 -e monit5
monit6  a xterm -name monit6 -e monit6
monit7  a xterm -name monit7 -e monit7
scnch   a xterm -name scnch  -e 'fsclient -n -w -s | grep /!\*scan_check..'
xterm   d xterm
monan   a xterm -name monan  -e monan
mona    d popen 'cd /tmp;rdbe30_mon.py -h 239.0.2.10 -p 20021 -H rdbea 2>&1' -n rdbemona
monb    d popen 'cd /tmp;rdbe30_mon.py -h 239.0.2.20 -p 20022 -H rdbeb 2>&1' -n rdbemonb
monc    d popen 'cd /tmp;rdbe30_mon.py -h 239.0.2.30 -p 20023 -H rdbec 2>&1' -n rdbemonc
mond    d popen 'cd /tmp;rdbe30_mon.py -h 239.0.2.40 -p 20024 -H rdbed 2>&1' -n rdbemond
```

TOW2023 - Seminar

FS Station Code

What does a station has to offer to the FS?
How to simplify access to data of the FS?
How to control your equipment from FS with “fsmonitor”?
How to command the FS in your code?
How to read answers from the FS in your code?
How to manually interact with the FS from remote?
How to use multiple devices in your FS?
How to integrate FS in your certified control loop?

How to use multiple devices in your FS?

Sample 2 DBBCs:

```
/usr2/control/dbbad.ctl
```

➔ **Commands**

```
...
dbbc=power=1
...
fila10g=arp off
...
```

```
/usr2/control/dbba2.ctl
```

➔ **Commands**

```
...
dbbc2=power=1
...
fila10g2=arp off
...
```

But what if you have more than the allowed standard devices?

Attention … here comes a hack!!!

How to use multiple devices in your FS?

But what if you have more than the allowed standard devices?

`socat` – Multipurpose relay

e.g. `socat TCP-LISTEN:142,fork,reuseaddr TCP:192.168.1.1:142`

Extend `station.prc`

```
define mydev3        00000000000x
sy=/usr2/st/bin/myscript_device3.sh > /dev/null 2> /dev/null
enddef
```

Script `myscript_device3.sh`

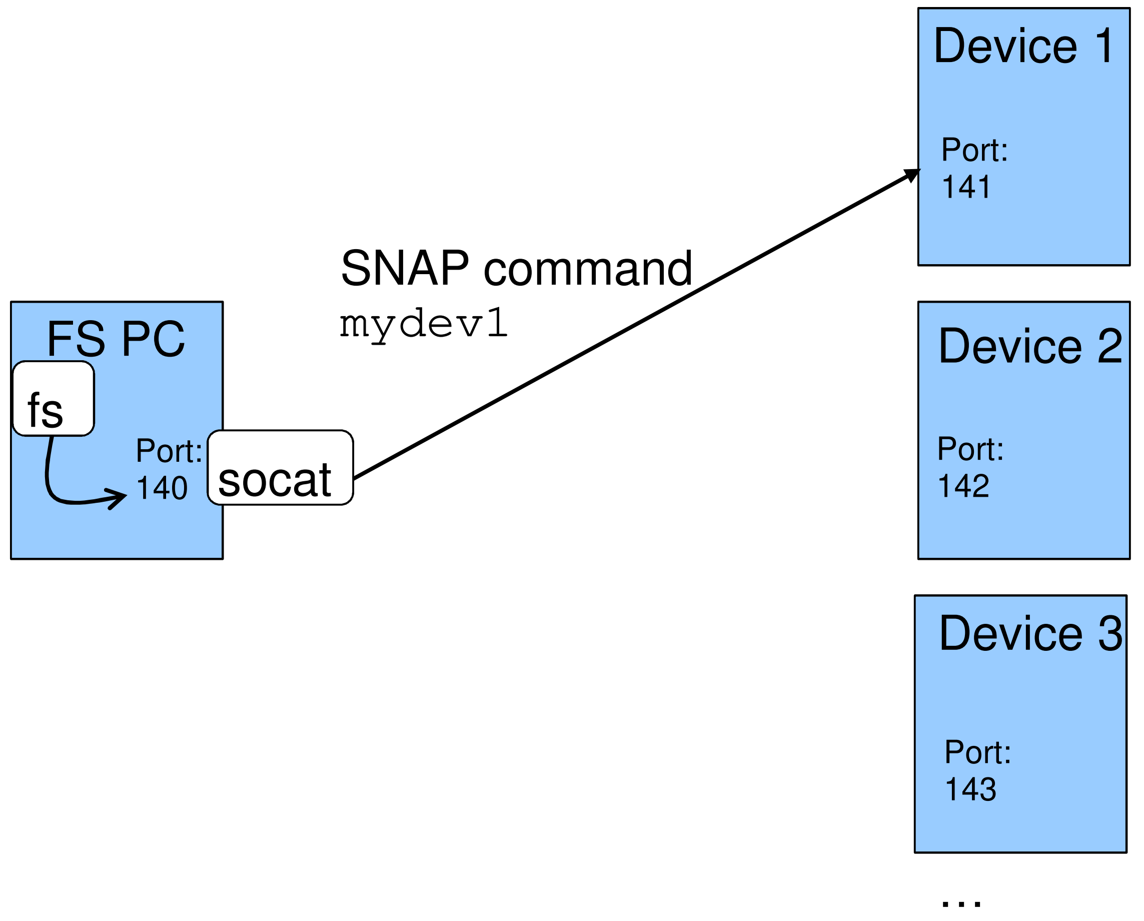
```
socat_basic_call="socat TCP-LISTEN:140,fork,reuseaddr TCP:"
# Kill previous patching
COMMAND="ps ax | grep \"${socat_basic_call}\" | grep -v grep | grep -o -E \"^[ ]*[0-9]+\" | tr '\n' ' '"
SOCAT_PIDS=`eval $COMMAND`
if [[ -n $SOCAT_PIDS ]]; then
    kill -9 $SOCAT_PIDS
    #echo -e "Killing processes with \"kill -9 $SOCAT_PIDS\""
fi
# Start patching of communication with socat
SOCAT_CALL="${socat_basic_call}192.168.1.1:143 > /dev/null 2> /dev/null &"
eval $SOCAT_CALL
```

How to use multiple devices in your FS?

But what if you have more than the allowed standard devices?

`socat` **– Multipurpose relay**

e.g. `socat TCP-LISTEN:142,fork,reuseaddr TCP:192.168.1.1:142`

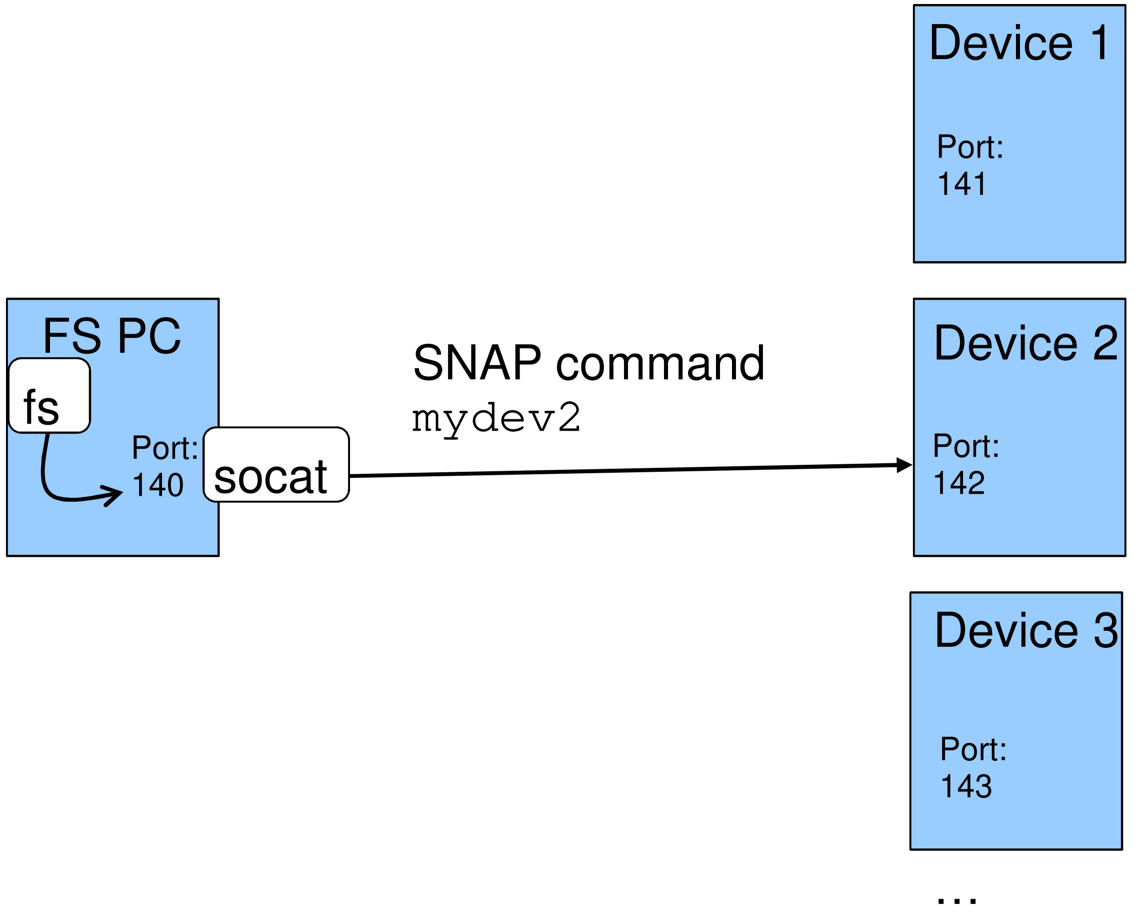


How to use multiple devices in your FS?

But what if you have more than the allowed standard devices?

`socat` – Multipurpose relay

e.g. `socat TCP-LISTEN:142,fork,reuseaddr TCP:192.168.1.1:142`

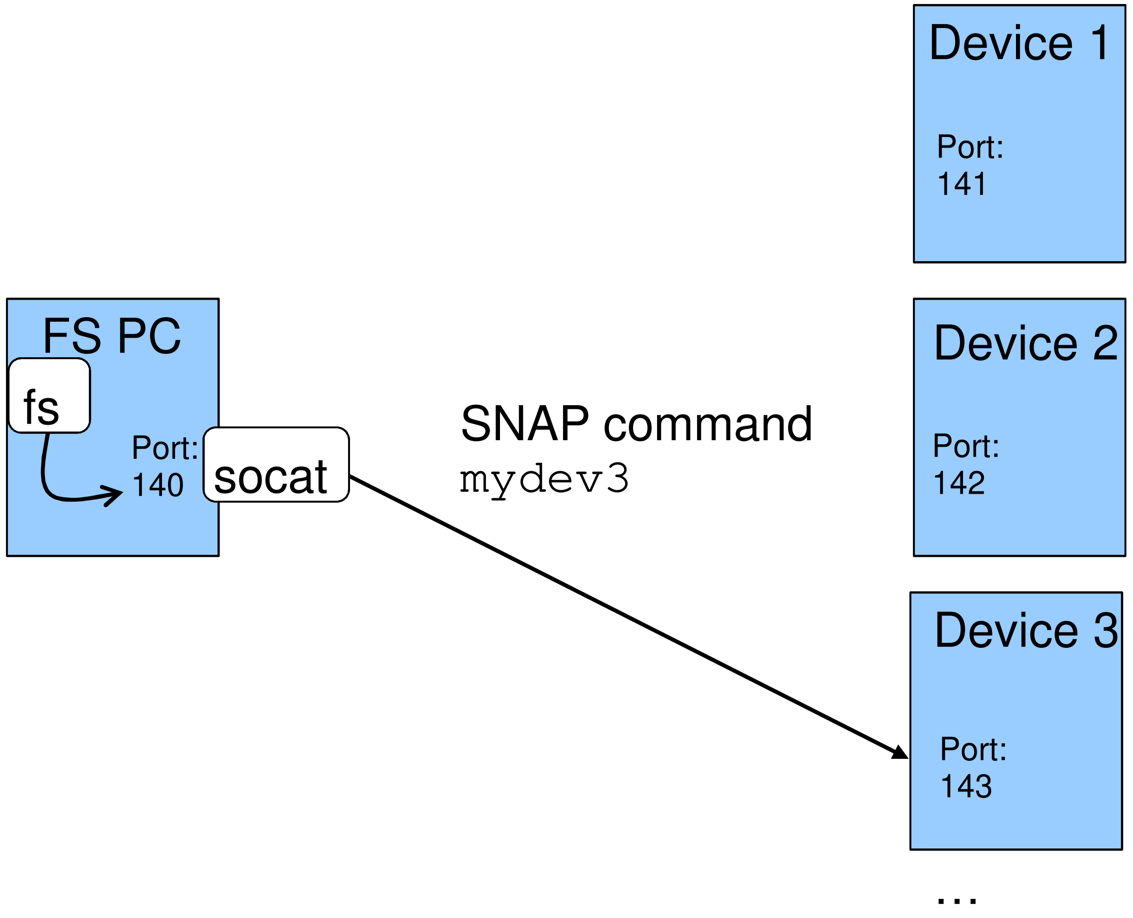


How to use multiple devices in your FS?

But what if you have more than the allowed standard devices?

`socat` – Multipurpose relay

e.g. `socat TCP-LISTEN:142,fork,reuseaddr TCP:192.168.1.1:142`

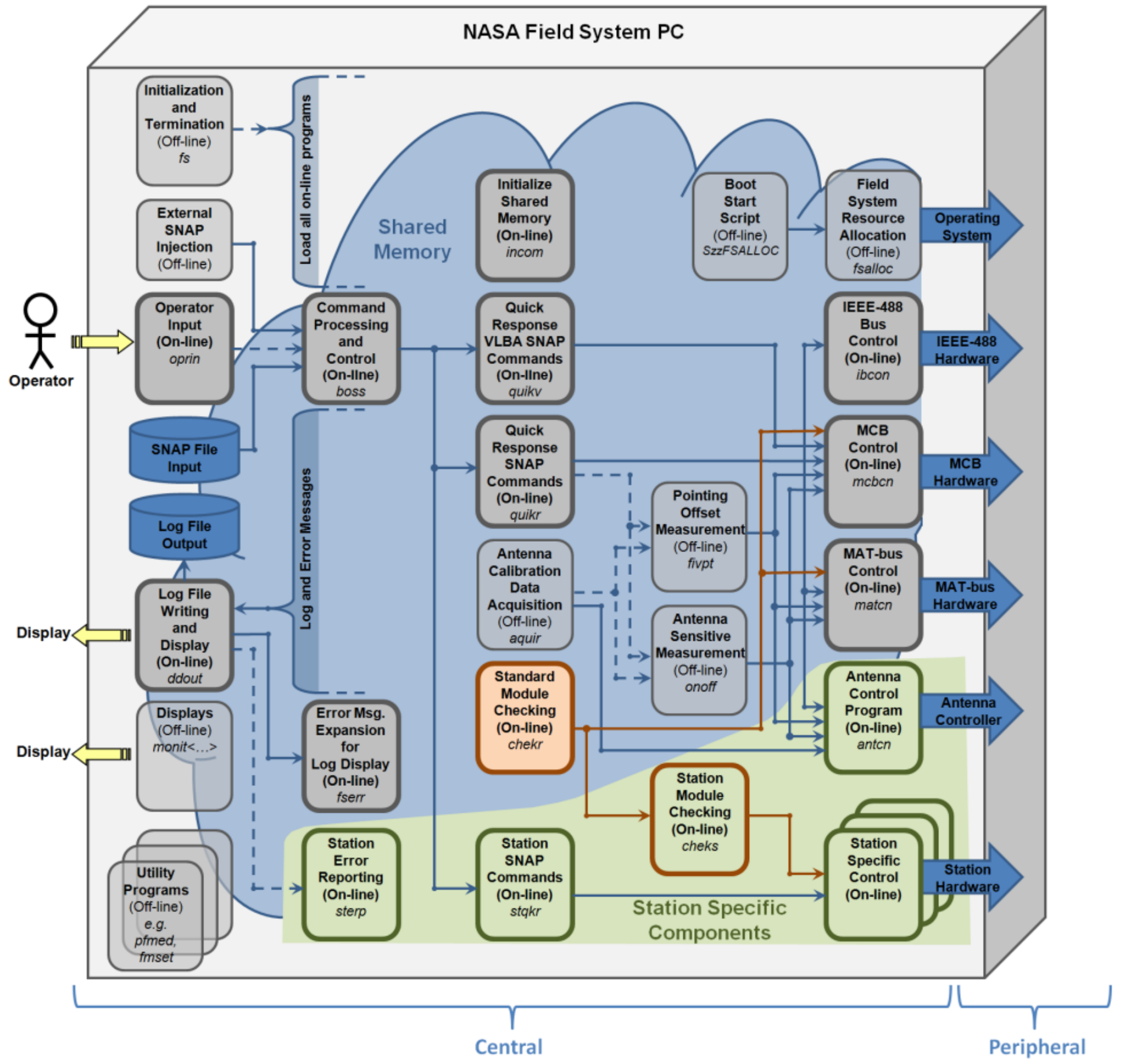


TOW2023 - Seminar

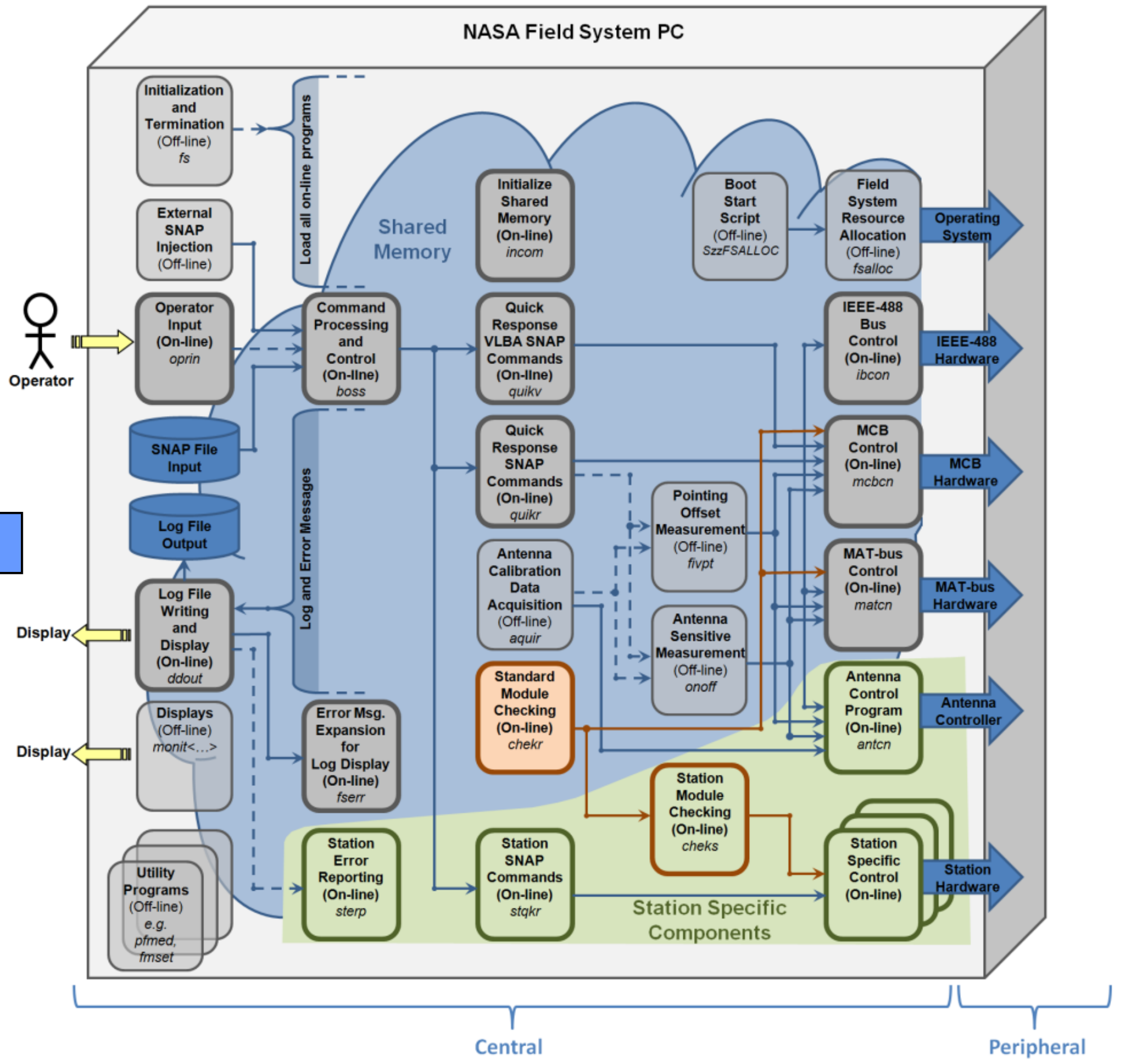
FS Station Code

What does a station has to offer to the FS?
How to simplify access to data of the FS?
How to control your equipment from FS with “fsmonitor”?
How to command the FS in your code?
How to read answers from the FS in your code?
How to manually interact with the FS from remote?
How to use multiple devices in your FS?
How to integrate FS in your certified control loop?

How to integrate FS in your certified control loop?


NASA Field System PC
Initialization and Termination (Off-line) fs
External SNAP Injection (Off-line)
Load all on-line programs
Shared Memory
Initialize Shared Memory (On-line) incom
Boot Start Script (Off-line) SzzFSALLOC
Field System Resource Allocation (Off-line) fsalloc
Operating System
Operator
Operator Input (On-line) oprin
Command Processing and Control (On-line) boss
Quick Response VLBA SNAP Commands (On-line) quikv
IEEE-488 Bus Control (On-line) ibcon
IEEE-488 Hardware
SNAP File Input
Log File Output
Quick Response SNAP Commands (On-line) quikr
Pointing Offset Measurement (Off-line) fivpt
MCB Control (On-line) mcbcn
MCB Hardware
Log and Error Messages
Antenna Calibration Data Acquisition (Off-line) aquir
Antenna Sensitive Measurement (Off-line) onoff
MAT-bus Control (On-line) matcn
MAT-bus Hardware
Display
Log File Writing and Display (On-line) ddout
Standard Module Checking (On-line) chekr
Antenna Control Program (On-line) antcn
Antenna Controller
Display
Displays (Off-line) monit<...>
Error Msg. Expansion for Log Display (On-line) fserr
Station Module Checking (On-line) cheks
Utility Programs (Off-line) e.g. pfmed, fmset
Station Error Reporting (On-line) sterp
Station SNAP Commands (On-line) stqkr
Station Specific Components
Station Specific Control (On-line)
Station Hardware
Central
Peripheral

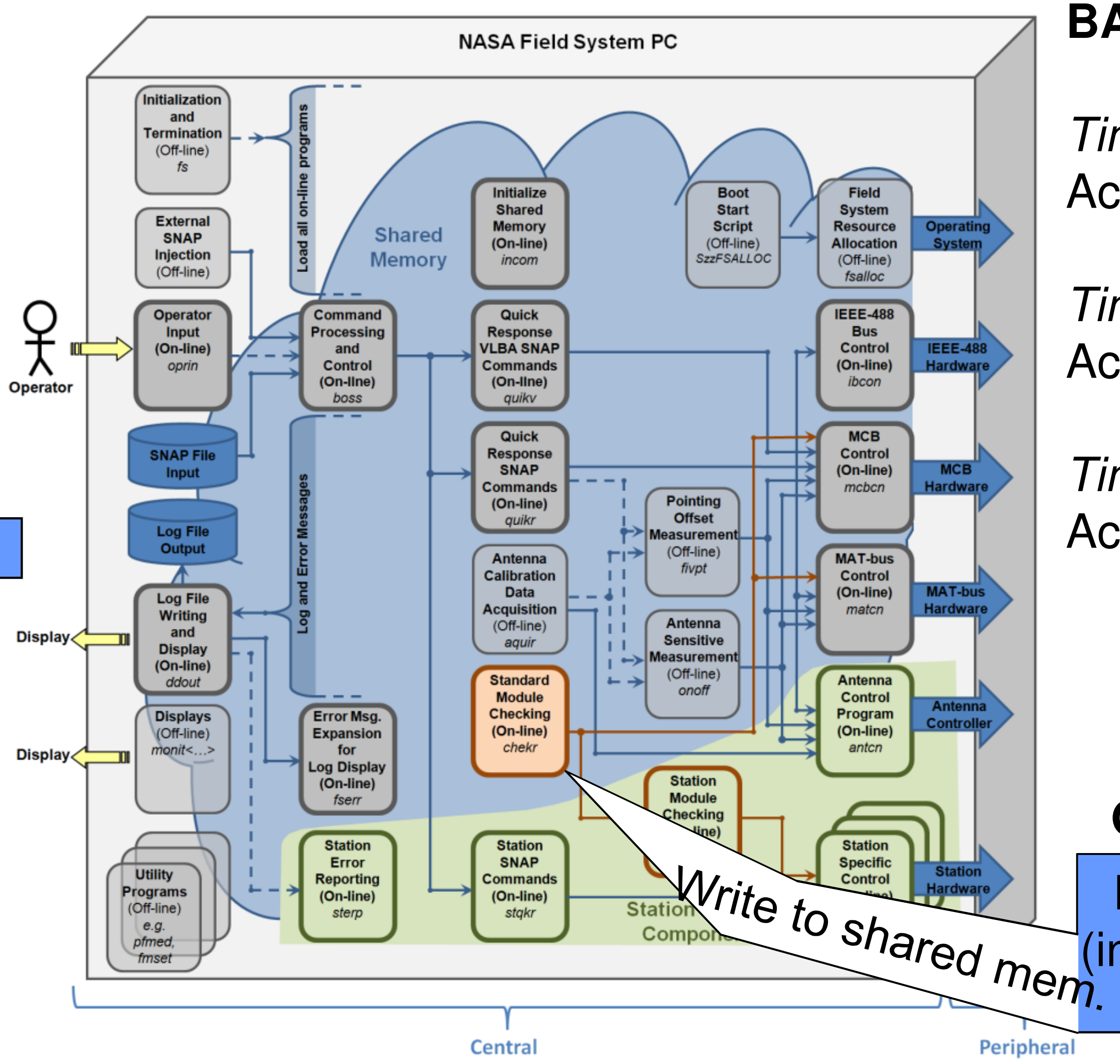
How to integrate FS in your certified control loop?


NASA Field System PC
Initialization and Termination (Off-line) fs
External SNAP Injection (Off-line)
Operator
Operator Input (On-line) oprin
SNAP File Input
Log File Output
Log File Writing and Display (On-line) ddout
Display
Displays (Off-line) monit<...>
Display
Utility Programs (Off-line) e.g. pfmed, fmset
Load all on-line programs
Log and Error Messages
Shared Memory
Command Processing and Control (On-line) boss
Error Msg. Expansion for Log Display (On-line) fserr
Station Error Reporting (On-line) sterp
Initialize Shared Memory (On-line) incom
Quick Response VLBA SNAP Commands (On-line) quikv
Quick Response SNAP Commands (On-line) quikr
Antenna Calibration Data Acquisition (Off-line) aquir
Standard Module Checking (On-line) chekr
Station SNAP Commands (On-line) stqkr
Boot Start Script (Off-line) SzzFSALLOC
Pointing Offset Measurement (Off-line) fivpt
Antenna Sensitive Measurement (Off-line) onoff
Station Module Checking (On-line) cheks
Station Specific Components
Field System Resource Allocation (Off-line) fsalloc
IEEE-488 Bus Control (On-line) ibcon
MCB Control (On-line) mcbcn
MAT-bus Control (On-line) matcn
Antenna Control Program (On-line) antcn
Station Specific Control (On-line)
Operating System
IEEE-488 Hardware
MCB Hardware
MAT-bus Hardware
Antenna Controller
Station Hardware
Central
Peripheral

Show current status to the user

Shared mem. specific, local GUI

How to integrate FS in your certified control loop?


NASA Field System PC
Initialization and Termination (Off-line) fs
Load all on-line programs
External SNAP Injection (Off-line)
Shared Memory
Initialize Shared Memory (On-line) incom
Boot Start Script (Off-line) SzzFSALLOC
Field System Resource Allocation (Off-line) fsalloc
Operating System
Operator
Operator Input (On-line) oprin
Command Processing and Control (On-line) boss
Quick Response VLBA SNAP Commands (On-line) quikv
IEEE-488 Bus Control (On-line) ibcon
IEEE-488 Hardware
SNAP File Input
Quick Response SNAP Commands (On-line) quikr
MCB Control (On-line) mcbcn
MCB Hardware
Log File Output
Log and Error Messages
Pointing Offset Measurement (Off-line) fivpt
Antenna Calibration Data Acquisition (Off-line) aquir
MAT-bus Control (On-line) matcn
MAT-bus Hardware
Display
Log File Writing and Display (On-line) ddout
Antenna Sensitive Measurement (Off-line) onoff
Standard Module Checking (On-line) chekr
Antenna Control Program (On-line) antcn
Antenna Controller
Display
Displays (Off-line) monit<...>
Error Msg. Expansion for Log Display (On-line) fserr
Station Module Checking
Station Error Reporting (On-line) sterp
Station SNAP Commands (On-line) stqkr
Station Specific Control
Station Hardware
Utility Programs (Off-line) e.g. pfmed, fmset
Station Components
Central
Peripheral

Write to shared mem.

Show current status to the user

Shared mem. specific, local GUI

BATCH-Mode

Timestamp
Action =>
<=Reply
Timestamp
Action =>
<=Reply
Timestamp
Action =>
<=Reply

Only if commanded

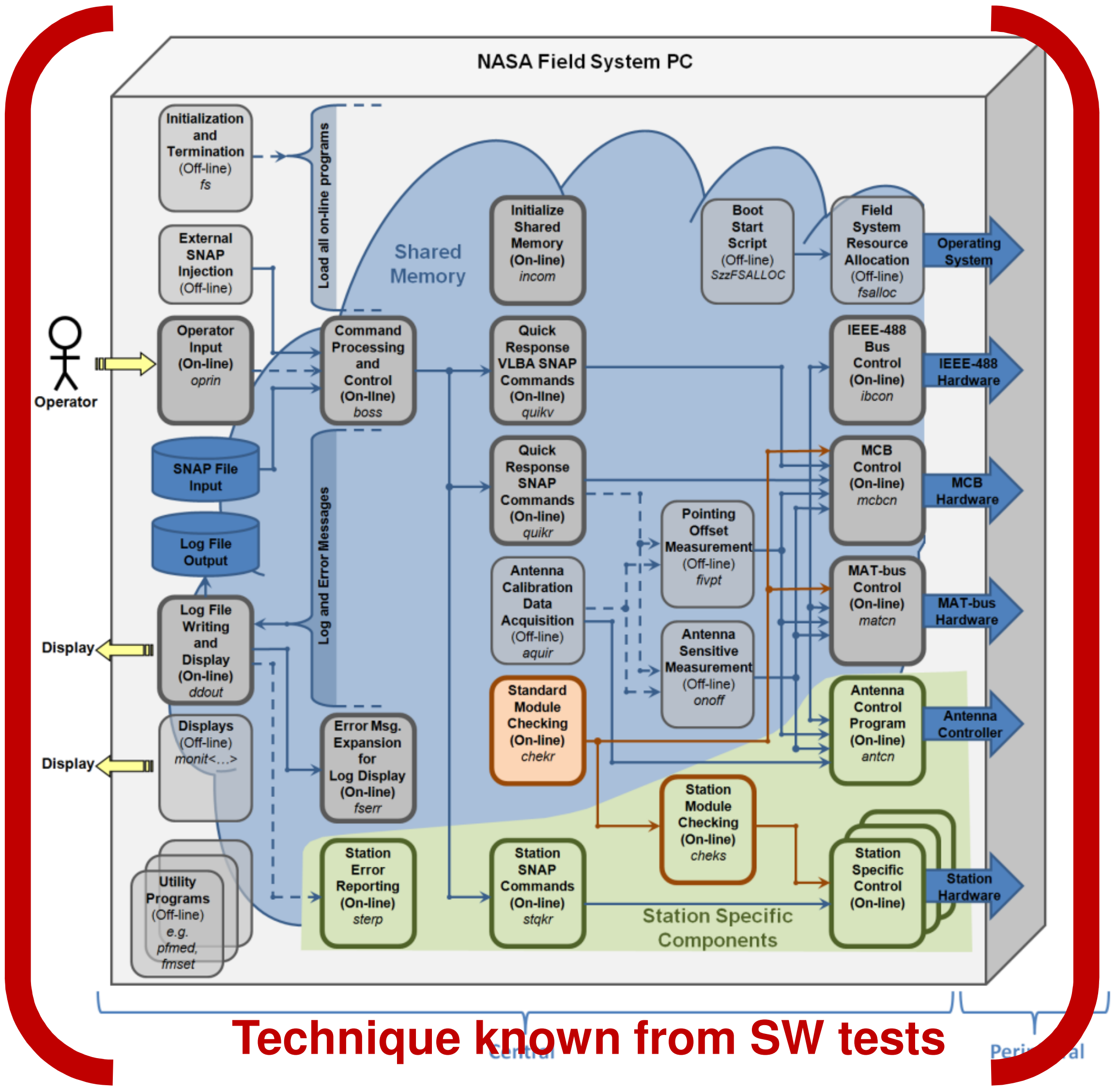
Own programs

Program (int. Timing)

Action =>
<=Reply

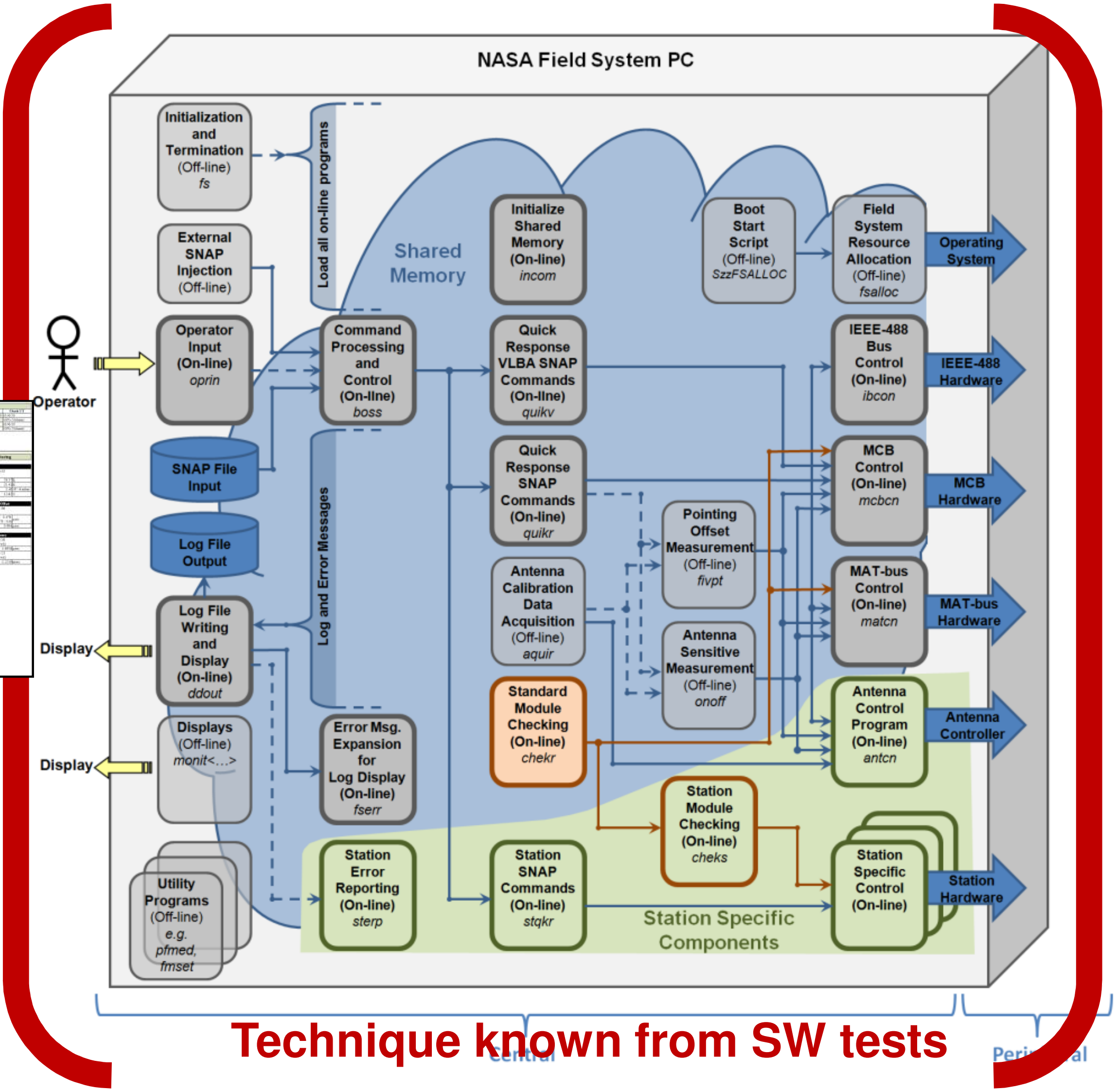
Specific, not monitored

How to integrate FS in your certified control loop?


NASA Field System PC
Initialization and Termination (Off-line) fs
External SNAP Injection (Off-line)
Operator Input (On-line) oprin
Operator
SNAP File Input
Log File Output
Log File Writing and Display (On-line) ddout
Display
Displays (Off-line) monit<...>
Display
Utility Programs (Off-line) e.g. pfmed, fmset
Load all on-line programs
Log and Error Messages
Shared Memory
Command Processing and Control (On-line) boss
Error Msg. Expansion for Log Display (On-line) fserr
Initialize Shared Memory (On-line) incom
Quick Response VLBA SNAP Commands (On-line) quikv
Quick Response SNAP Commands (On-line) quikr
Antenna Calibration Data Acquisition (Off-line) aquir
Standard Module Checking (On-line) chekr
Boot Start Script (Off-line) SzzFSALLOC
Pointing Offset Measurement (Off-line) fivpt
Antenna Sensitive Measurement (Off-line) onoff
Station Module Checking (On-line) cheks
Field System Resource Allocation (Off-line) fsalloc
IEEE-488 Bus Control (On-line) ibcon
MCB Control (On-line) mcbcn
MAT-bus Control (On-line) matcn
Antenna Control Program (On-line) antcn
Station Specific Control (On-line)
Operating System
IEEE-488 Hardware
MCB Hardware
MAT-bus Hardware
Antenna Controller
Station Hardware
Station Error Reporting (On-line) sterp
Station SNAP Commands (On-line) stqkr
Station Specific Components
Central
Peripheral

Technique known from SW tests

How to integrate FS in your certified control loop?

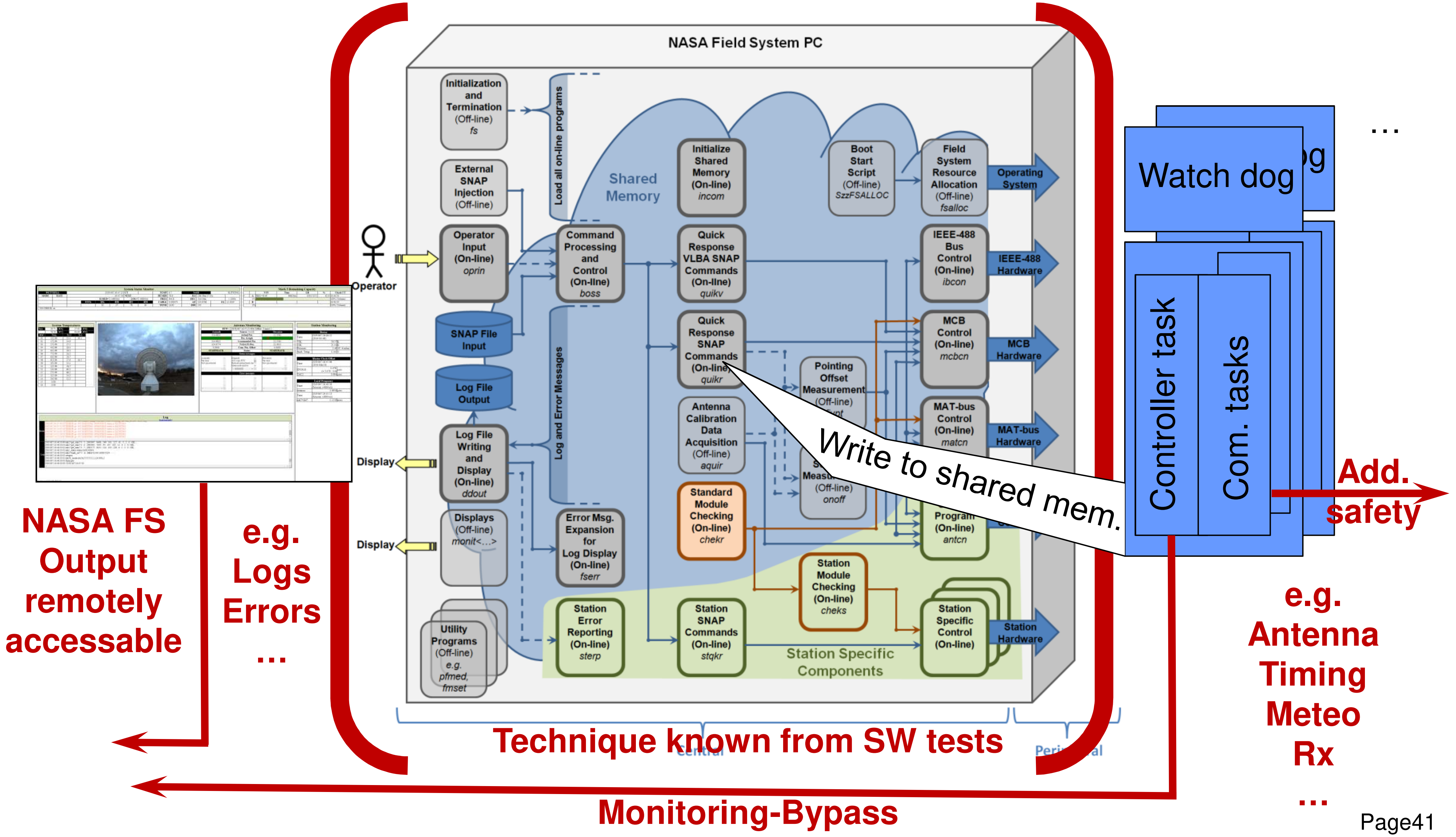

NASA Field System PC
Initialization and Termination (Off-line) fs
External SNAP Injection (Off-line)
Load all on-line programs
Shared Memory
Initialize Shared Memory (On-line) incom
Boot Start Script (Off-line) SzzFSALLOC
Field System Resource Allocation (Off-line) fsalloc
Operating System
Operator
Operator Input (On-line) oprin
Command Processing and Control (On-line) boss
Quick Response VLBA SNAP Commands (On-line) quikv
IEEE-488 Bus Control (On-line) ibcon
IEEE-488 Hardware
SNAP File Input
Log File Output
Log and Error Messages
Quick Response SNAP Commands (On-line) quikr
Pointing Offset Measurement (Off-line) fivpt
MCB Control (On-line) mcbcn
MCB Hardware
Antenna Calibration Data Acquisition (Off-line) aquir
MAT-bus Control (On-line) matcn
MAT-bus Hardware
Display
Log File Writing and Display (On-line) ddout
Antenna Sensitive Measurement (Off-line) onoff
Standard Module Checking (On-line) chekr
Antenna Control Program (On-line) antcn
Antenna Controller
Display
Displays (Off-line) monit<...>
Error Msg. Expansion for Log Display (On-line) fserr
Station Module Checking (On-line) cheks
Utility Programs (Off-line) e.g. pfmed, fmset
Station Error Reporting (On-line) sterp
Station SNAP Commands (On-line) stqkr
Station Specific Components
Station Specific Control (On-line)
Station Hardware
Central
Peripheral

NASA FS Output remotely accessable

e.g. Logs Errors …

Technique known from SW tests

How to integrate FS in your certified control loop?


NASA Field System PC
Initialization and Termination (Off-line) fs
External SNAP Injection (Off-line)
Load all on-line programs
Shared Memory
Initialize Shared Memory (On-line) incom
Boot Start Script (Off-line) SzzFSALLOC
Field System Resource Allocation (Off-line) fsalloc
Operating System
Operator
Operator Input (On-line) oprin
Command Processing and Control (On-line) boss
Quick Response VLBA SNAP Commands (On-line) quikv
IEEE-488 Bus Control (On-line) ibcon
IEEE-488 Hardware
SNAP File Input
Log File Output
Quick Response SNAP Commands (On-line) quikr
Pointing Offset Measurement (Off-line)
MCB Control (On-line) mcbcn
MCB Hardware
Log and Error Messages
Antenna Calibration Data Acquisition (Off-line) aquir
MAT-bus Control (On-line) matcn
MAT-bus Hardware
Display
Log File Writing and Display (On-line) ddout
Standard Module Checking (On-line) chekr
(Off-line) onoff
Program (On-line) antcn
Display
Displays (Off-line) monit<...>
Error Msg. Expansion for Log Display (On-line) fserr
Station Module Checking (On-line) cheks
Utility Programs (Off-line) e.g. pfmed, fmset
Station Error Reporting (On-line) sterp
Station SNAP Commands (On-line) stqkr
Station Specific Control (On-line)
Station Hardware
Station Specific Components
Central
Peripheral
Write to shared mem.
Watch dog
...
Controller task
Com. tasks
Add. safety
NASA FS Output remotely accessable
e.g. Logs Errors ...
Technique known from SW tests
e.g. Antenna Timing Meteo Rx ...
Monitoring-Bypass

How to integrate FS in your certified control loop?

```
#ifndef __PERIODIC_CLOUDCOVERAGE_THREAD__
#define __PERIODIC_CLOUDCOVERAGE_THREAD__

#include "meteo_simple_thread.hpp"
#include "meteo_simple_semvar.hpp"

class PeriodicCloudCoverageThreadClass : public meteo_CSimpleThread
{
    private:
        meteo_semvar<char>                priv_cProtectPrivateVariablesSemaphore;
        bool                              priv_bInitSuccessful;
        std::string                       priv_strUsedIPAddress;
        std::string                       priv_strUsedWebPageURL;
        unsigned long                     priv_ulSamplingTimeInSec;
        bool                              priv_bError;
        unsigned short                    priv_usFatalError;
        bool                              priv_bThreadIsRunning;
        bool                              priv_bSilentMode;
        unsigned long                     priv_ulTimeOfLastCompleteDataset;
                                          /// priv_ulTimeOfLastCompleteDataset = Time stamp of the last
                                          /// completely and correctly read meteo data set
        double                            priv_dCloudCoverage;
        bool                              priv_bUseWebpageImage;

    public:
        enum ErrorCodesEnum {PERIODICCLOUDCOVTHREAD_OK = 0,
                             PERIODICCLOUDCOVTHREAD_NOK,
                             PERIODICCLOUDCOVTHREAD_NOKINITALREADYDONE,
                             PERIODICCLOUDCOVTHREAD_NOKNOTREADY,
                             PERIODICCLOUDCOVTHREAD_FATALSEMAPHORE,
                             PERIODICCLOUDCOVTHREAD_FATALHTTPADDRESS,
                             PERIODICCLOUDCOVTHREAD_FATALSYSTEMCALL,
                             PERIODICCLOUDCOVTHREAD_FATALSVALUEFILE
                            };
    private:
        int iRun (bool bDetached=THREAD_DETACHED, const int iDelayMillis=0 )
        {
            return meteo_CSimpleThread::iRun (bDetached, iDelayMillis);
        }
        void * pvEntry ();
        void vSleep (unsigned short usSleepTimeMillisec);
```

How to integrate FS in your certified control loop?

```
    public:
        PeriodicCloudCoverageThreadClass ();
        PeriodicCloudCoverageThreadClass (const PeriodicCloudCoverageThreadClass & CIn);
        ~PeriodicCloudCoverageThreadClass ();
        PeriodicCloudCoverageThreadClass & operator= (const PeriodicCloudCoverageThreadClass & CIn);
        unsigned short usRun (const std::string & strUsedIPAddress,
                              const std::string & strUsedDefaultWebPageURL,
                              const unsigned long & ulSamplingTimeInSec,
                              const bool & bSilentMode = false);
        unsigned short usRunWebImageUse (const std::string & strUsedWebPageURL,
                                         const unsigned long & ulSamplingTimeInSec,
                                         const bool & bSilentMode = false);
        bool bIsErrorAndReset();
        bool bIsError();
        bool bIsFatalError();
        bool bIsRunning();
        unsigned short usGetCloudCoverage (unsigned long & ulTimeOfLastCompleteDataset,
                                           double & dCloudCoveragePercent);
        void vPrintCloudCoverageValues ();
};

#endif //__PERIODIC_CLOUDCOVERAGE_THREAD__
```

Important parts are
- usRun => Initialize and startup of the thread
- pvEntry => Loop with frequently performed tasks

How to integrate FS in your certified control loop?

usRun => Initialize and startup of the thread

```
unsigned short PeriodicCloudCoverageThreadClass::usRun (const std::string & strUsedIPAddress,
                      const std::string & strUsedDefaultWebPageURL,
                      const unsigned long & ulSamplingTimeInSec,
                      const bool & bSilentMode)
{
    unsigned short usRetVal = PERIODICCLOUDCOVTHREAD_OK;
    unsigned long ulBlockHandle = 0;

    /// Block semaphores
    try
    {
        priv_cProtectPrivateVariablesSemaphore.vBlock (ulBlockHandle);
    }
    catch (...)
    {
        priv_usFatalError = PERIODICCLOUDCOVTHREAD_FATALSEMAPHORE;
        return PERIODICCLOUDCOVTHREAD_NOK;
    }

    if (priv_bInitSuccessful)
    {
         usRetVal = PERIODICCLOUDCOVTHREAD_NOKINITALREADYDONE;
    }
    else
    {
        /// Copy IP address and timeout and init connection
        priv_bUseWebpageImage = false;
        priv_ulSamplingTimeInSec = ulSamplingTimeInSec;
        priv_strUsedIPAddress = strUsedIPAddress.c_str();
        priv_strUsedWebPageURL = strUsedDefaultWebPageURL.c_str();
        priv_bInitSuccessful = true;
    }

    /// Unblock semaphores
    try
    {
        priv_cProtectPrivateVariablesSemaphore.vUnblock (ulBlockHandle);
    }
    catch (...)
    {
        priv_usFatalError = PERIODICCLOUDCOVTHREAD_FATALSEMAPHORE;
        return PERIODICCLOUDCOVTHREAD_NOK;
    }
```

Definition DEF2.17. of «semaphore» (see (Singhal and Shivaratri 1994, *l.c. page 16 f.*) and (Stevens 1990, *l.c. page 148 f.*))
A (binary) semaphore is an integer variable with two atomic operations to synchronize concurrent threads or processes: «test&set» (decrement) the variable to block and «release» (increment) it to unblock a critical section. Only the task, which successfully sets and blocks the semaphore is allowed to operate the critical section. Each semaphore also has a queue, in which the blocked requesters are waiting for access.

Semaphore

Set and define configuration values

How to integrate FS in your certified control loop?

```
    /// Check if init failed
    if (usRetVal != PERIODICCLOUDCOVTHREAD_OK)
    {
        return usRetVal;
    }

    /// Start thread
    if (meteo_CSimpleThread::iRun())
    {
        /// Block semaphores
        try
        {
            priv_cProtectPrivateVariablesSemaphore.vBlock (ulBlockHandle);
        }
        catch (...)
        {
            priv_usFatalError = PERIODICCLOUDCOVTHREAD_FATALSEMAPHORE;
            return PERIODICCLOUDCOVTHREAD_NOK;
        }
        priv_bInitSuccessful = false;
        /// Unblock semaphores
        try
        {
            priv_cProtectPrivateVariablesSemaphore.vUnblock (ulBlockHandle);
        }
        catch (...)
        {
            priv_usFatalError = PERIODICCLOUDCOVTHREAD_FATALSEMAPHORE;
            return PERIODICCLOUDCOVTHREAD_NOK;
        }
        return PERIODICCLOUDCOVTHREAD_NOK;
    }

    return PERIODICCLOUDCOVTHREAD_OK;
}
```

Start periodic loop as thread

How to integrate FS in your certified control loop?

pvEntry => Loop with frequently performed tasks

```
void * PeriodicCloudCoverageThreadClass::pvEntry ()
{
    /// Local variables
    ...
while (1)
    {
        ulBlockHandle = 0;
        ulStarttime = time(NULL);

        /// =============================================================
        /// Get private variables
        /// =============================================================
        /// Block semaphores
        try
        {
            priv_cProtectPrivateVariablesSemaphore.vBlock (ulBlockHandle);
        }
        catch (...)
        {
            priv_usFatalError = PERIODICCLOUDCOVTHREAD_FATALSEMAPHORE;
            goto SleepManagementContinue;
        }

        /// Copy current values from private to local
        bError = false;
        usFatalError = priv_usFatalError;
        ulSleepTimeSec = priv_ulSamplingTimeInSec;
        strUsedIPAddress = priv_strUsedIPAddress.c_str();
        strUsedWebPageURL = priv_strUsedWebPageURL.c_str();
        bSilentMode = priv_bSilentMode;
        bUseWebpageImage = priv_bUseWebpageImage;

        /// Unblock semaphores
        try
        {
            priv_cProtectPrivateVariablesSemaphore.vUnblock (ulBlockHandle);
        }
        catch (...)
        {
            priv_usFatalError = PERIODICCLOUDCOVTHREAD_FATALSEMAPHORE;
            goto SleepManagementContinue;
        }

        if (usFatalError != PERIODICCLOUDCOVTHREAD_OK)
        {
            goto SleepManagementContinue;
        }
```

Prepare configuration values

How to integrate FS in your certified control loop?

```
        /// =============================================================
        /// Read from hardware
        /// =============================================================
...

ErrorContinue:
        /// =============================================================
        /// Set private variables
        /// =============================================================
        /// Block semaphores
        try
        {
            priv_cProtectPrivateVariablesSemaphore.vBlock (ulBlockHandle);
        }
        catch (...)
        {
            priv_usFatalError = PERIODICCLOUDCOVTHREAD_FATALSEMAPHORE;
            goto SleepManagementContinue;
        }
        priv_bError = bError;
        priv_usFatalError = usFatalError;
        if (!bError && usFatalError == PERIODICCLOUDCOVTHREAD_OK)
        {
            priv_dCloudCoverage = dCloudCoverage;
            priv_ulTimeOfLastCompleteDataset = usTimeOfLastCompleteDataset;
        }
        if (usFatalError != PERIODICCLOUDCOVTHREAD_OK)
        {
            priv_bError = true;
        }

        /// Set state as running
        priv_bThreadIsRunning = true;


        /// Unblock semaphores
        try
        {
            priv_cProtectPrivateVariablesSemaphore.vUnblock (ulBlockHandle);
        }
        catch (...)
        {
            priv_usFatalError = true;
            goto SleepManagementContinue;
        }
```

Fetched values
from hardware

Store fetched values

How to integrate FS in your certified control loop?

```
SleepManagementContinue:
        /// Sleep management
        ulEndtime = time(NULL);
        if (ulSleepTimeSec > 0)
        {
            if (ulEndtime - ulStarttime >= ulSleepTimeSec)
            {
                continue;
            }
            else
            {
                ulSleepTimeSec = ulSleepTimeSec - (ulEndtime - ulStarttime);
            }
            sleep (ulSleepTimeSec);
        }
        else
        {
            struct timespec STime;
            STime.tv_sec = 0;
            STime.tv_nsec = 40000000; // 40 msec
            nanosleep (&STime, NULL);
        }
    }

    return NULL;
}
```

Timing of
control loop

TOW2023 - Seminar

FS Operations

Thank you …