Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

Swiss Confederation

Federal Department of Home Affairs FDHA
**Federal Office of Meteorology and Climatology MeteoSwiss**

# Fieldextra

*A Fortran program to manipulate gridded fields*

Jean-Marie Bettems

# Fieldextra – Identity card (1)

**Some examples of typical model pre- / post-processing tasks**:

Pre-proc

- merge surface temperature from IFS over sea and from COSMO over land to produce a single field suited for the assimilation cycle
- interpolate Swiss radar composite onto the COSMO-2 grid for feeding the latent heat nudging process
- interpolate KENDA analysis from regular grid to ICON triangular grid to start re-forecasts for R&D

Post-proc

- compute air density on a set of pressure and height above ground levels, interpolated on a geographical lat/lon grid, in GRIB 2
- compute EPS probabilities from COSMO-LEPS members
- create a *single* XML file with time series of parameters from COSMO-2 / -7 / -EPS and IFS for a set of locations
- create a bitmap for the condition ‘probability of 6h sum of total precipitation exceeding 25mm is larger than 0’

**Use fieldextra for any of these tasks… but much more is possible !**

# Fieldextra – Identity card (2)

## Generic tool to process model data and gridded observations

- propose a large set of primitive operations, which can be freely combined and iterated (**toolbox**)
- **import** GRIB 1, GRIB 2, NetCDF and some special simple ASCII format
- offer best **compatibility** with COSMO(-ART), ICON(-ART) and IFS models
- implemented in **Fortran 2008**, as a single but modular code
- a **control file**, a collection of Fortran namelists, defines the set of operations to apply on the input data

# Fieldextra – Identity card (3)

## Primary focus is the production environment

- **robustness** of the code
- catching and reporting of run time **exceptions**
- careful processing of **field meta-information**, with systematic checks to avoid meaningless products
- consistent treatment of **field undefined values** along processing chain
- memory and CPU **optimization**
- capability to produce multiple different products reading input data only once (**IO optimization**)
- comprehensive **diagnostic** and **profiling**
- support **inter-process** communication (parallel production suite)

## Supported by extensive regression suite

# Fieldextra – Identity card (4)

- **Portable** code
  - Only **standard Fortran** features
  - Should work on **any UNIX / Linux / Mac platform**
    - *Not ported on Window*
    - *Nested OpenMP mode may be problematic on some platforms*
  - Tested on multiple platforms, with multiple compilers

- **Documented** code
  - More than 10k lines of external documentation
  - Fieldextra Primer, User manual, FAQ…
  - Large set of commented examples (cookbook)

- **Community support**
  - cosmo-fieldextra@cosmo-model.org

# Fieldextra – Identity card (5)

- **Licenced software**,
  - free to all COSMO members.
  - free licences are available for the **R&D community**, but without support.

- **Availability**
  - **Master code repository on GitHub**
    https://github.com/COSMO-ORG/fieldextra (private repository)
  - **Package on COSMO web site**
    http://www.cosmo-model.org/content/support/software/default.htm
  - **Full installation at ECMWF on cca** (UNIX group cfxtra)
    /perm/ms/ch/ch7/projects/fieldextra
  - **Full installation at CSCS on tsa and daint** (UNIX group s83c)
    /project/s83c/fieldextra/{tsa,daint}

# Fieldextra – Identity card (6)

- Usage
  - **Pre-processing** tool and **non-graphical production tool** at MeteoSwiss
    - Upscaling of KENDA analysis, preparation of COSMO LBC using ICON pollen fields, preparation of forcing fields for INCA ...
    - Thousands of products generated each day based on KENDA, COSMO-1E, COSMO-2E, IFS-HRES, IFS-ENS, IFS-SEAS, INCA
  - EPS derived fields from **ICON-D2-EPS**, **ICON-EPS**, derivation of products for verification and for German flight control at DWD
  - Other **operational users at ARPAE, IMS, REMET and RHM**
  - **COSMO-LEPS** production at ECMWF

- Official **COSMO software** for post-processing

# Fieldextra – Identity card (7)

- Development started in **May 1998**
  - **Original idea by** Pirmin Kaufman / MeteoSwiss
  - **Lead developer** is Jean-Marie Bettems / MeteoSwiss
  - More than **16 FTE** invested up to now
  - With contributions from Felix Ament, Axel Barleben, **Petra Baumann**, Philipp Glatt, Christophe Hug, Pirmin Kaufmann, Guy de Morsier, Donat Perler, Florian Prill, Anne Roches, Vanessa Stauch, Martin Schraner, Balazs Szintai, André Walser, Tanja Weusthoff

- About 180k lines of standard **Fortran 2008**
  - **Actively developed code** (about 2 release per year)
  - **Well documented code**, about 20% are comment lines
  - Linked with own **GRIB1** library and with the **ICON tools** library from DWD
  - Linked with the following **external libraries**:
    RTTOV (observation operator for satellite), ECMWF ecCodes (GRIB2),
    JasPer (JPEG in GRIB2), libaec (entropy coding in GRIB 2),
    netCDF library (NetCDF), hdf5 and zlib library (support NetCDF)
  - OpenMP implementation for **shared memory parallelism**

# Features (1)

## Data import

- GRIB 1 and GRIB 2, including files with mixed records
- NetCDF
- BLK_TABLE (special simple ASCII format)
- screening based on field name, level, temporal identity, EPS identity …
- conditional extraction (e.g. T<0 and QC>0)
- supported **data sources**
  - COSMO
  - ICON
  - IFS-HRES, IFS-ENS, IFS-SEAS (on regular grid)
  - other NWP (with restricted functionality)
  - Swiss radar composite
- supported **data representation**
  - geographical lat/lon, rotated lat/lon
  - kilometric grids (swiss coordinates, italian coordinates)
  - ICON triangular grid

# Features (2.1)

## Data manipulation *(some are restricted to regular grids)*

- Modification of meta-information (both global and local)
- Manipulation of undefined values
- Change of reference system (vector fields)
- Change of units (scale, offset)
- Field normalization
- Other mathematical operators (polynoms, logarithm, exponential…)
- Conditional manipulation of field values
- Other local transformations based using external information (height correction, kalman correction, MOS estimation...)
- Lateral re-gridding (interpolation operators)
- Lateral smoothing (convolution operators, controlled by a mask)
- Lateral reduction (coarser grid)
- Aggregation for specified regions (avg, min, max, count, quantile, fit)
- Temporal interpolation (filling time series gaps)
- Other temporal operators (delta, sum, avg, min, max, date_of_max ... )

# Features (2.2)

## Data manipulation (ctn’d)

- Definition of an arbitrary height or p surface (condition on auxiliary field)
- Vertical interpolation (from model levels to pressure, height, theta, pv)
- Vertical interpolation (from model levels to an arbitrary surface)
- Vertical extrapolation (constant, constant gradient, hydrostatic)
- Vertical stretching, vertical translation (to a different topography)
- Vertical extremum (between two arbitrary surfaces)
- Vertical integral (between two arbitrary surfaces)
- Smoothed vertical gradient (between two arbitrary surfaces)
- Vertical derivative
- Conditional merge of multiple fields
- Comparison of 2 fields (difference, normed difference, ...)
- Creation of composed ensembles (e.g. lagged EPS)

# Features (3)

## Computation of new fields

- Based on standard meteorological formulae
  *wind direction & velocity, vertical wind shear, CFL criteria,*
  *geostrophic wind, geostrophic vorticity, 3D wind divergence,*
  *potential vorticity on model surfaces,*
  *dew point, mixing ratio, enthalpy, wet bulb temperature,*
  *percentage of total precipitation in snow,*
  *CAT indices, stability indices, soil moisture index …*

- Based on look-up tables
- Based on MOS estimator (incl. gridded field and probability information)
- AdaBoost classifier and derived probability
- Neighbourhood probability
- EPS probability, perturbation, quantile, interquantile, stdev and mean (weighted or not)

# Features (4)

**Data subset** *(some are restricted to regular grids)*

- grid points or geographical locations
- rectangular subdomain, dense or sparse (incl. region envelope)
- frames
- vertical slices

**Data export**

- NetCDF (geog. and rotated lat/lon grid, and ICON triangular grid)
- GRIB 1 (geog. and rotated lat/lon grid)
- GRIB 2 (geog. and rotated lat/lon grid, and ICON triangular grid)
- CSV, tabulated
- XML (template based)
- pseudo-ANETZ (‘meteograms’)
- clients specific
- … with many output characteristics are controlled by the user (missing value code, precision, verbosity, etc.)

# Features (5.1)

## Possibility to combine and iterate any previously described operations

- **Example**

  *Predictant for freezing rate, based on COSMO-7:*

  (1) find the regions where there is a temperature inversion, with the temperature being below 0C at the surface and reaching above 0C in the troposphere

  (2) in these regions, compute the normalized integral of the temperature between the first and the second 0C levels

  (3) produces a single NetCDF file with the following fields:

  + thickness of the ‘warm’ region in (1), NA otherwise<br>
  + normalized integral (2), NA outside of (1)<br>
  + temperature on the lowest level

# Features (5.2)

```
&Process
  in_type="INCORE"
  out_file="road_forecast.nc"
  out_type="NETCDF"
  out_type_nodegeneratedim=.TRUE.
/
&Process in_field = "HFL", levmin=20, levmax=60 /
&Process in_field = "HHL", levmin=20, levmax=61 /

&Process
  in_file="lfff00000000"
  out_file="road_forecast.nc"
  out_type="NETCDF"
  out_type_nodegeneratedim=.TRUE.
/
&Process in_field = "T", levmin = 20, levmax = 60 /
```

**Collect data from COSMO-7**

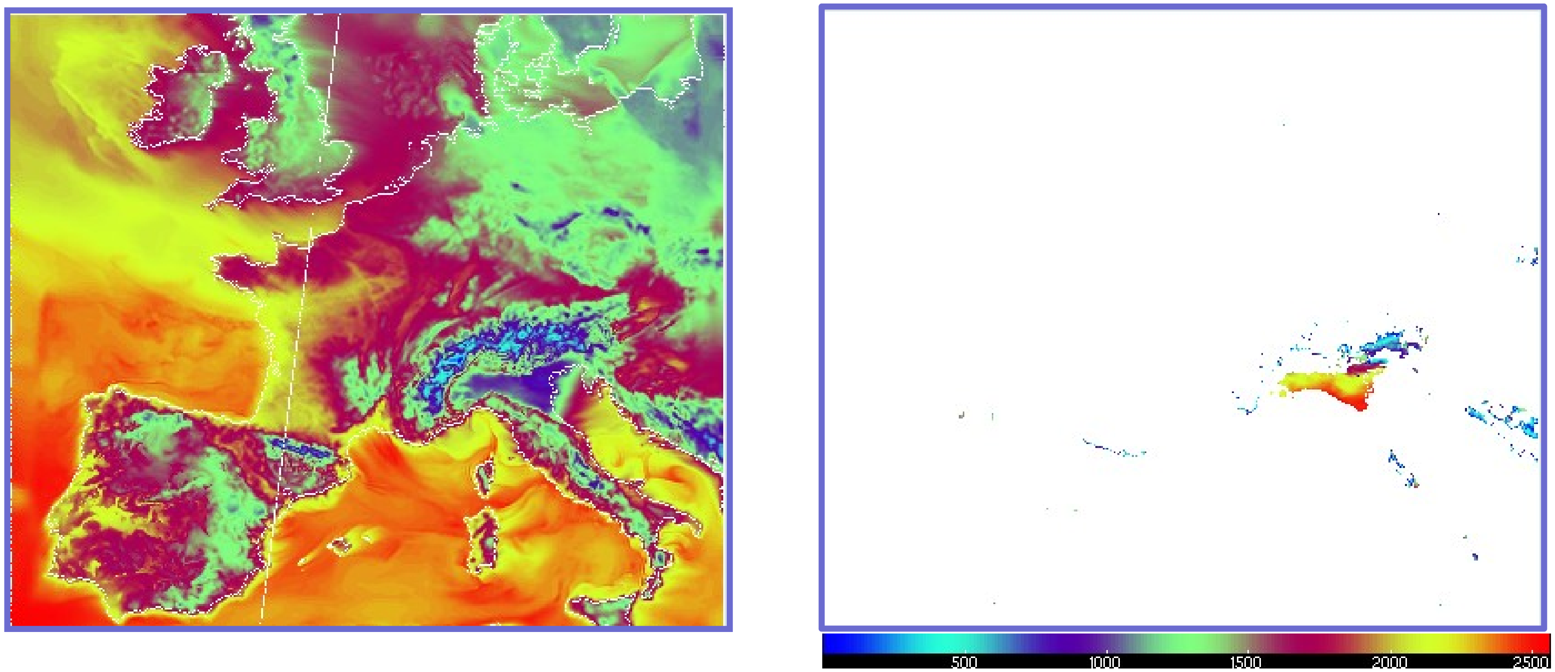
```
&Process tmp1_field = "T" /
&Process tmp1_field = "HHL" /
&Process tmp1_field = "HFL", tag = "ml_height" /
&Process tmp1_field = "HFL", levlist = 60, tag = "height_lowest_level" /
&Process tmp1_field = "HFL", tag = "height_first_0deg",
                voper = "find_condition,T=273.15,height_lowest_level,up" /

&Process tmp2_field = "T" /
&Process tmp2_field = "HHL" /
&Process tmp2_field = "ml_height" /
&Process tmp2_field = "ml_height", tag = "height_second_0deg",
                voper = "find_condition,T=273.15,height_first_0deg,up" /
&Process tmp2_field = "height_first_0deg" /

&Process tmp3_field = "HHL" /
&Process tmp3_field = "ml_height" /
&Process tmp3_field = "height_first_0deg" /
&Process tmp3_field = "height_second_0deg" /
&Process tmp3_field = "T", levlist=60, tag = "t_lowest_level" /
&Process tmp3_field = "height_second_0deg" , tag='thickness_warm_layer',
                  poper='sum,__self__+height_first_0deg*-1' /
&Process tmp3_field = "T", tag = "integral_t_high",
               voper = "integ_z2z,height_first_0deg,height_second_0deg",
               poper='mask,t_lowest_level>273.15'                        /

&Process out_field = "t_lowest_level" /
&Process out_field = "integral_t_high" /
&Process out_field = "thickness_warm_layer",
            poper='mask,t_lowest_level>273.15' /
```

**Iterative processing**

# Features (5.3)

**NetCDF output produced by fieldextra.**

Lowest temperature (left panel) and thickness in [m] between first and second 0C levels (right panel).



# Features (6)

## Robustness

- Comprehensive automated **regression suite**
- Regular and comprehensive **tests** for a large spectrum of applications, with different models on different platforms, by a community of users
- Comprehensive run time **consistency checks**
  (about 5000 programmed exceptions)
- Consequent handling of **missing values** (data, meta-information)

## Optimization

- Read **input** once, produce as many output files as defined
- Shared memory **nested parallelism**
  (parallel import, concurrent output production,
  parallel computation of operators)
- **Time critical mode**, for concurrent run of model and post-processing
- Build-in **code profiling** (CPU, memory)

# Features (7)

## Extensibility (beyond build-in functionalities)

- Code functionality can **in some cases** be easily extended by writing a single new procedure
  - *operator to compute a new field*
  - *product specific post-processing*
  - *specific output format*
- Other types of functionalities are fairly easy to add
  - *support of new types of regular grids*
  - *support of new meta-information*
  - *introduction of new operators (poper, toper, voper, hoper)*
- Due to the **modular and abstract nature of the code**, unforeseen features at the time of the code design can still be implemented in a coherent way (at least up to this release)…
  **… but deep knowledge of the code is required!**

# Features (8)

## User interface

- **Control file** composed of Fortran namelists
  - *Namelists reflect the internal logic of the code*
  - *Setting is a **complex manual task***
  - *Many commented examples are provided (**cookbook**) to mitigate this issue*
  - *For some common functionalities, simple wrapper scripts, with a limited set of options, are provided (**fx tools**) to mitigate this issue*

- **External resources** as flat ASCII files
  - *Dictionaries for field short names and characteristics*
  - *Location file for definition of geographical locations*
  - *Region file for definition of geographical regions*
  - *Slice file for definition of vertical slices paths*
  - *Template files for TMPL_BASE output ('forms')*
  - *Coefficient files for AdaBoost, Kalman and MOS corrections*

# Strengths and Weaknesses

- Very robust (no crash triggered by input singularities)
- Safe (product consistency)
- Versatile (toolbox, code extensibility)
- Well documented (code and external documentation)
- Efficient, both in terms of time to solution and memory footprint

- Steep learning curve (unfriendly user interface)
- Limited development capacity (limited pool of knowledgable people)
- Modular but single and complex code

**Fieldextra is primary a production tool, and not a tool for quick manipulations in daily workflow (except for fx tools)**

# Learning to use fieldextra

- See **what is possible** by peeking at
  - *cookbook/README.cookbook*

- **Study** the fieldextra primer document
  - *documentation/1_FirstContact.pdf*

- **Understand** (some) of the examples in the cookbook directory, refers to the README.user for more detailed information
  - *cookbook/*.nl*
  - *documentation/README.user*

- **Find and adapt** similar cookbook example matching your need, use fieldextra@cosmo-model.org to ask for community support

# Fieldextra credits

***Original idea:***

Pirmin Kaufmann / MeteoSwiss

***Lead developer:***

Jean-Marie Bettems / MeteoSwiss

***Core team:***

Petra Baumann / MeteoSwiss

Jean-Marie Bettems / MeteoSwiss

***Contributions from:***

Felix Ament / Uni Hamburg (horizontal re-gridding)

Mathias Aschwanden / MeteoSwiss (upscaling, tools)

Axel Barleben / DWD (operator EDP and others)

Philipp Glatt / MeteoSwiss (GRIB2 support)

Christophe Hug / MeteoSwiss (IFS, GME, vert. coord.)

Pirmin Kaufmann / MeteoSwiss (neighbourhood prob)

Guy de Morsier / MeteoSwiss (some operators)

Donat Perler (AdaBoost, many indices)

Florian Prill / DWD (icontools)

Anne Roches (Vorticity on p-surfaces)

Martin Schraner (OpenMP parallellization, environment)

Vanessa Stauch / MeteoSwiss (SIA2028 product)

Balazs Szintai / MeteoSwiss (turbulence operators)

Andre Walser / MeteoSwiss (EPS derived products)

Tanja Weusthoff / MeteoSwiss (ASCII output)

***External consultants:***

CSCS (profiling, Intel compiler, grib1)

# Enjoy fieldextra!

```
!+****************************************************************************
SUBROUTINE generate_output(multi_pass_mode, just_on_time, last_call,           &
              datacache, data_origin, tot_nbr_input,                &
              out_paths, out_types, out_modes,                      &
              out_grib_keys, out_spatial_filters,                   &
              out_subset_size, out_subdomain, out_gplist, out_loclist, &
              out_data_reduction, out_postproc_modules,             &
              nbr_gfield_spec, gen_spec, ierr, errmsg              )
!=============================================================================
!
! Root procedure to generate output files
!
!-----------------------------------------------------------------------------
 ! Dummy arguments
 LOGICAL, INTENT(IN)                                :: multi_pass_mode     ! Multiple pass mode?
 LOGICAL, DIMENSION(:), INTENT(IN)                  :: just_on_time        ! True if prod. now
 LOGICAL, INTENT(IN)                                :: last_call           ! True if last call
 CHARACTER(LEN=*), INTENT(IN)                       :: datacache           ! Data cache file
 TYPE(ty_fld_orig), INTENT(IN)                      :: data_origin         ! Data origin
 INTEGER, DIMENSION(:), INTENT(IN)                  :: tot_nbr_input       ! Expected nbr. input
 CHARACTER(LEN=*), DIMENSION(:), INTENT(IN)         :: out_paths           ! Output files names
 TYPE(ty_out_spec), DIMENSION(:), INTENT(IN)        :: out_types           !   types
 TYPE(ty_out_mode), DIMENSION(:), INTENT(IN)        :: out_modes           !   modes
 INTEGER, DIMENSION(:,:), INTENT(IN)                :: out_grib_keys       !   grib specs
 INTEGER, DIMENSION(:), INTENT(IN)                  :: out_subset_size     !   subset size
 INTEGER, DIMENSION(:,:), INTENT(IN)                :: out_subdomain       !   subdomain definition
 INTEGER, DIMENSION(:,:,:), INTENT(IN)              :: out_gplist          !   gp definition
 CHARACTER(LEN=*), DIMENSION(:,:), INTENT(IN)       :: out_loclist         !   locations definition
 CHARACTER(LEN=*), DIMENSION(:), INTENT(IN)         :: out_spatial_filters ! Condition defining filter
 TYPE(ty_out_dred), DIMENSION(:), INTENT(IN)        :: out_data_reduction  ! Data reduction spec
 CHARACTER(LEN=*), DIMENSION(:), INTENT(IN)         :: out_postproc_modules ! Specific postprocessing
 INTEGER, DIMENSION(:,:), INTENT(IN)                :: nbr_gfield_spec     !+ specifications of
 TYPE(ty_fld_spec_root), DIMENSION(:), INTENT(IN)   :: gen_spec            !+ fields to generate
 INTEGER, INTENT(OUT)                               :: ierr                ! error status
 CHARACTER(LEN=*), INTENT(OUT)                      :: errmsg              !   error message

 ! Local parameters
 CHARACTER(LEN=*), PARAMETER          :: nm='generate_output: ' ! Tag

 ! Local variables
 LOGICAL                              :: exception_detected, exception, use_postfix
 LOGICAL                              :: unique_ftype, multiple_grid, exist
 LOGICAL, DIMENSION(3*mx_iteration+1) :: tmp_fddata_alloc, tmp_gpdata_alloc
 LOGICAL, DIMENSION(3*mx_iteration+1) :: tmp_value_alloc, tmp_flag_alloc
 INTEGER                              :: i1, i2, i3, i_fd, i_vd
 INTEGER                              :: nbr_input
 INTEGER                              :: out_idx, ios, idx_vd_defined
 CHARACTER(LEN=strlen)                :: messg, temporal_res, out_path
 TYPE(ty_fld_type)                    :: out_ftype


 ! Initialize variables
 !---------------------
 ierr = 0  ;  errmsg = ''
 exception_detected = .FALSE.
 tmp_fddata_alloc(:) = .FALSE.  ;  tmp_gpdata_alloc(:) = .FALSE.
 tmp_value_alloc(:) = .FALSE. ; tmp_flag_alloc(:) = .FALSE.


 ! Create/update data cache file
 !------------------------------------------------------------------------
 ! The cache file must reflect the state of data(:) after the last call to
 ! collect_output (i.e. before any field manipulation done in prepare_pout)
```

```
! Loop over each output file
!---------------------------
output_file_loop: &
DO i1 = 1, nbr_ofile
 out_idx = data(i1)%ofile_idx
 nbr_input = COUNT( data(i1)%ifile_used )

 ! Skip bogus output
 IF ( data(i1)%ofile_bogus ) CYCLE output_file_loop
 ! Skip completed output
 IF ( data(i1)%ofile_complete ) CYCLE output_file_loop
 ! Skip empty data array
 IF ( ALL(.NOT. data(i1)%defined) ) CYCLE output_file_loop
 ! Only prepare output when all possible associated data have been collected
 ! or when 'just on time' production is active
 IF ( .NOT. last_call                        .AND.                  &
      nbr_input < tot_nbr_input(out_idx) .AND.                      &
      .NOT. just_on_time(out_idx)            ) CYCLE output_file_loop

 ! At this point the corresponding output file will be produced
 ! Keep track of completed output file
 IF ( nbr_input >= tot_nbr_input(out_idx) ) data(i1)%ofile_complete = .TRUE.

 ! Build name of output, considering a possible temporary postfix
 use_postfix = .FALSE.
 IF ( LEN_TRIM(out_postfix) /= 0 .AND. data(i1)%ofile_usepostfix .AND.    &
      .NOT. (data(i1)%ofile_firstwrite .AND. data(i1)%ofile_complete)  )  &
                                     use_postfix = .TRUE.
 out_path = out_paths(out_idx)
 IF ( use_postfix ) out_path = TRIM(out_path) // out_postfix

 ! Release memory allocated in previous call to prepare_pout (if any)
 DO i2 = 1, 3*mx_iteration+1
  IF ( tmp_value_alloc(i2) ) DEALLOCATE(data_tmp(i2)%values, data_tmp(i2)%defined)
  IF ( tmp_flag_alloc(i2) ) DEALLOCATE(data_tmp(i2)%flag)
  IF ( tmp_fddata_alloc(i2) ) THEN
   DEALLOCATE(data_tmp(i2)%field_type, data_tmp(i2)%field_origin,        &
data_tmp(i2)%field_name, data_tmp(i2)%field_grbkey,        &
         data_tmp(i2)%field_trange,                              &
         data_tmp(i2)%field_level, data_tmp(i2)%field_ltype,        &
         data_tmp(i2)%field_prob, data_tmp(i2)%field_epsid,         &
         data_tmp(i2)%field_vref, data_tmp(i2)%field_ngrid,         &
         data_tmp(i2)%field_scale, data_tmp(i2)%field_offset,       &
         data_tmp(i2)%field_vop, data_tmp(i2)%field_vop_usetag,     &
         data_tmp(i2)%field_vop_nlev, data_tmp(i2)%field_vop_lev,   &
         data_tmp(i2)%field_pop, data_tmp(i2)%field_hop,            &
         data_tmp(i2)%field_top, data_tmp(i2)%nbr_level,            &
         data_tmp(i2)%level_idx, data_tmp(i2)%nbr_eps_member,       &
         data_tmp(i2)%eps_member_idx, data_tmp(i2)%field_idx        )
  ENDIF
  IF ( tmp_gpdata_alloc(i2) ) THEN
   DEALLOCATE(data_tmp(i2)%gp_coord, data_tmp(i2)%gp_idx,                &
         data_tmp(i2)%gp_lat, data_tmp(i2)%gp_lon, data_tmp(i2)%gp_h)
  ENDIF
 END DO

 ! Prepare data for print out (calculate new fields, ... ; populate data_pout)
 ! * Info message
 IF ( just_on_time(out_idx) ) THEN
  messg = ' (just on time output)'
 ELSE IF ( nbr_input >= tot_nbr_input(out_idx) ) THEN
  messg = ' (all associated input collected)'
 ELSE
  messg = ''
 ENDIF
```