IMSYS Version 6.2 (98-11-08)

This is a "small" image handling system called IMSYS written in C.

IMSYS is a collection of more than 44 small programs centred on a common image interface. An image consists of a **text** header file located in the header directory (HDRPATH) and a **binary** data file located in the image directory (IMGPATH). The data file is addressed indirectly through the header file. The header file contains FITS-like keywords. The image name is the header file-name without the ".HDR" extension.

A new feature makes it possible to select arguments from the command line of a previous invocation of the same program. It is implemented by reading the old arguments from an argument file (one for each program) located in the argument directory (ARGS). If ':' appears as an argument it is replaced by the corresponding argument of the previous invocation of the program (if possible). If '!' is given as an argument it signifies that the remaining arguments should be taken from the saved command line starting at the corresponding position.

A short description of how to call a program can be obtained by typing the program name with no parameters. Note, that image names may **not** contain any **punctuation** characters. The following is a list of existing programs with a short description (parameters in [ ] are optional, several possible options are separated by |). Parameters following a single letter with a '–' prefix (Unix style) are also optional (e.g. –**x param** or –**xparam**).

1. **list -l -s -d levdist [image1] [image2] [image3]...**
   List directory information on images. –**l** produces a long listing.
   –**s** produces a short listing.
   –**d levdist** selects the images according to how equal they are to the given name **image1** by using the "Levenstein distance" between two strings. All images with a distance closer than or equal to **levdist** are included. You may also use wild card *(s) in **image1**, but not when using the the –**d** option, and remember to put " around any parameters containing *(s): list "ngc*".
   Note, this is not necessary under MS-DOS (PC).

2. **addpsf -t -s image Xc Yc A1 A2 R1 R2**
   Adds a **PSF** of the following form to an existing image:
   $$PSF(x,y) = \frac{A_1}{R_1^2} e^{-\pi\left((x-X_c)^2+(y-Y_c)^2\right)/R_1^2} + \frac{A_2}{R_2^2} e^{-\pi\left((x-X_c)^2+(y-Y_c)^2\right)/R_2^2}$$
   –**t** indicates that the Optical Transfer Function (the Fourier transform of the **PSF**) should be added instead:
   $$OTF(x,y) = \frac{A_1}{N^2} e^{-\pi\left((x-X_c)^2+(y-Y_c)^2\right)R_1^2} + \frac{A_2}{N^2} e^{-\pi\left((x-X_c)^2+(y-Y_c)^2\right)R_2^2}$$
   where we have assumed that the image has dimension N $\times$ N.
   –**s** indicates that the PSF should be subtracted instead of being added.

3. **addstar -g pow image top Xc Yc fwhm**
   Adds a Pseudo-Gaussian star with hight **top** and full width at half maximum **fwhm** at

location (**Xc**, **Yc**). Random noise is added.
–g pow indicates that the radial exponent **pow** should be different from 2. Default is a Gaussian (**pow** = 2).

4. **adgalaxy image Xc Yc Ie Re [PA] [eps]**
   Adds a de Vaucouleurs galaxy at the location (**Xc**, **Yc**). The surface brightness is **Ie** at the effective radius **Re**. **PA** is the position angle (in deg) of the major axis, and **eps** is the ellipticity. Random noise is added.

5. **aphot -n -e epdn image Xc Yc RadMax SkyMin SkyMax [RadMin]**
   Makes aperture photometry of an object centred on (**Xc**, **Yc**). All pixels with **RadMin** $\leq$ radius $\leq$ **RadMax** are sky subtracted and summed. The mean sky value is derived from pixels within the annulus **SkyMin** $\leq$ radius $\leq$ **SkyMax**. **–n** indicates that no sky subtraction should be done. **–e epdn** is used to assign the number of electrons per Digital Number. The default is **epdn** = 2.6. If **RadMin** is not given RadMin = 0 is assumed.

6. **badcol image Xc dX1 dX2 dX3 [Ymin] [Ymax]**
   Removes a bad column between rows **Ymin:Ymax** by linear interpolation of pixel values in the interval [**Xc–dX1, Xc+dX1**]. The endpoints of this interpolation are derived by taking the mean values for the two intervals [**Xc–dX3**, **Xc–dX2**] and [**Xc+dX2**, **Xc+dX3**]. This is done for each row. **Only** short (type = 2) images are allowed.

7. **bias -m<pts> image1 image2 [Bmin] [Bmax] [Xmin] [Xmax] [type] [Ymin] [Ymax]**
   Calculates and subtracts an individual bias level for each row as derived from the bias strip between columns **Bmin:Bmax**. **image2** contains the bias-subtracted image between columns **Xmin:Xmax** and rows **Ymin:Ymax**. The input image **image1** must be of type short. The output image **image2** can be any of the types short, float, long, and complex. The bias for row y is derived as the mean of values in the interval [**y–pts,y+pts**] as indicated by the option **–m<pts>**. If **pts** is negative one single average bias is subtracted from all rows.

8. **calc image1 op image2 [factor]**
   op = { { + | **add** | **ADD** } | { - | **sub** | **SUB** } | { * | **mul** | **MUL** } |
   { / | **div** | **DIV** } | { = | **EQ** }}
   calc image1 "/" image2 10000 <=> image1 = image1 / image2 * 10000
   It may be necessary to place quotes around **op**. If this does not work, use the the keywords indicated (e.g. **div**, **mul** etc). **'='** cannot be used with High-C under MS-DOS (use **eq** or **EQ**).

9. **const image op c.r [c.i]**
   op = {{ + | **add** | **ADD** } | { - | **sub** | **SUB** } | { * | **mul** | **MUL** } | { / | **div** | **DIV** } | { @ | **sqrt** | **SQRT** } | { = | **eq** | **EQ** } | { > | **rsh** | **RSH** } | { < | **lsh** | **LSH** }}
   const image + 500 <=> image = image + 500 ">" and "<" are used for right and left shift, respectively. It is necessary to place quoates around **op**, ">" or "<". It is possible to multiply and divide type 1, 2 and 4 images by float constants in order to give

images the same mean level. The square root of an image can now be taken by using the @ operator. Keywords can also be used. '=' cannot be used with the High-C compiler under MS-DOS. **const image @ 10 <=> image = 10 * sqrt(image).**

10. **convpsf -b FltSub image A1 A2 R1 R2**
    Convolves the (N × N) **image** with a **PSF** defined by

    $$PSF(x,y) = \frac{A_1}{R_1^2} e^{-\pi(x^2+y^2)/R_1^2} + \frac{A_2}{R_2^2} e^{-\pi(x^2+y^2)/R_2^2}$$

    N must be a power of 2. The result is stored in **image**.
    –**b FltSub** gives a value which will replace **BLANK** pixels.

11. **cosmic -x -y -g gain -n noise -s err -k kap -r npt -i nit -m met img img1**
    Detects and replaces cosmic ray events in **img** by fitting a sum of all polynomial terms in (x, y) up to the 3rd order plus a 4th order term in x, y, or radius (r) to a 5x5 square region centred on any suspected spike (the central pixel is by default omitted from the fit). A pixel is suspected to be a spike if it is greater than the maximum of the 4 means of opposite pairs of pixels plus the standard deviation. The fit is repeated after having omitted the **one** pixel with the largest residual. This rejection procedure can be repeated several times. **-x** indicates that a 4th order term in **x** is used. **-y** indicates that a 4th order term in **y** is used. **-xy** indicates that a 4th order term in **r** is used (default). –**g gain** is used to assign the CCD gain in electrons per ADU. –**n noise** is used to assign the CCD rms read noise in electrons. –**s err** gives the CCD scale error as a fraction. –**k kap** is a rejection factor (**kap*sigma** criterion). –**r npt** is the max. nr. of pixels that can be rejected in the fit. –**i nit** gives the number of detection passes through the image. –**m met** indicates the spacing between pixels in a 5x5 data matrix. –**m1:** Spacing = 1 pixel. –**m2:** Spacing = 1.41 pixel (45 deg. rotation). –**m3:** Spacing = 2 pixels. If the residual of the central pixel is larger than **kap*sigma** the pixel value is replaced with the fitted value. The modified image is placed in **img1**.
    NOTE, the image data must be oversampled (**PSF-FWHM** > 5 pixels), and it might be an advantage to use a sample spacing of 1.41.

12. **embed image1 image2**
    Embeds **image1** in upper left corner of existing **image2**

13. **fft -i -b FltSub image1 image2**
    Performs a Fast Fourier Transform of **image1**. The full complex result is placed in **image2**. **image1** can have any type, but it must have dimensions which are powers of 2. –**i** indicates an inverse **FFT**. –**b FltSub** gives a value which will replace **BLANK** pixels.

14. **gauss -x -y -m -e epdn image Xc Yc RadMax SkyMin SkyMax**
    2-dim (–**xy** or no option) or 1-dim (–**x** or –**y** option) Gaussian fitting program. If –**xy** or no option is present the program fits a two dimensional single Gaussian function to a stellar image approximately located at pixel (**Xc**, **Yc**). If, in addition, the –**m** option is present, a sum of 2 Gaussians are fitted. The Gaussian(s) is (are) fitted to pixels with

radii < **RadMax**. The sky level is calculated from pixels inside the ring: **SkyMin** < radius < **SkyMax**. If option **–x** is present it fits a one dimensional Gaussian function to the line obtained by summing the lines between **Yc–RadMax** and **Yc+RadMax**. The Gaussian is fitted to pixels between **Xc–RadMax** and **Xc+RadMax**. The sky level is calculated from pixels with **SkyMin** < **abs(x–Xc)** < **SkyMax**, and with **Yc–SkyMax** < y < **Yc+SkyMax**. If option **–y** is present the one dimensional Gaussian fitting is done along the y-axis instead. **–e epdn** is used to assign the number of electrons per Digital Number used to calculate the weights. If **–e epdn** is **not** given, equal weights are used.

15. **getkey image**
    (test program which reads FITS keywords from image)

16. **icp image1 image2**
    Copy **image1** to new **image2**

17. **ifpix image rop const aop value**
    rop = { { < | **lt** | **LT** } | {=| **eq** | **EQ** } | { > | **gt** | **GT** }}
    aop = { { + | **add** | **ADD** } | { - | **sub** | **SUB** } | { * | **mul** | **MUL** } |
    { / | **div** | **DIV** } | { = | **eq** | **EQ** }}
    ifpix image "<" 500 = 1000 <=> if (image < 500) image = 1000
    It may be necessary to place quotes around **rop**, "<", ">" etc. In case of difficulties, use the keywords. It is not possible to use "=" with the High-C compiler under MS-DOS (use **eq** or **EQ**). It is possible to change a **BLANK** value to another value by **ifpix image eq const eq value**

18. **imake -p gain -n sdev -s seed image value npix nlin** [**type**]
    image = **value** + [Gaussian noise]
    npix: X-dimension
    nlin: Y-dimension
    type: Image type = { 1 | 2 | 3 | 4 | 5 } (default: 2)
    –p gain adds photon noise with mean **gain * value**.
    –n sdev adds Gaussian noise with standard deviation **sdev**.
    –s seed changes the seed for the random generator.
    If **seed** is not given, the system time is used as seed.

19. **irm -i image1 [image2] [image3]...**
    Removes a list of images from system. If **–i** is given it asks for confirmation by a prompt: **image: ? (y/n/e)** for every image in the list. **y:** Yes, **n:** No, and **e:** Exit. Other letters mean no. You may use wild card *(s) in **image1:** irm "ng*" **Note**, this is not necessary under DOS.

20. **ishift -x -y image X0 Y0**
    Performs a cyclic shift of columns and rows in **image**. (0, 0) $\longrightarrow$ **(X0, Y0)**. The **–x** option indicates that the x-dimension is mirror imaged. The **–y** option indicates that the y-dimension is mirror imaged.

21. **kingfit -n nmax -o image1 image Sky Ic Rc Xc Yc Rmax**
Fits a King profile of the form

$$f(x,y) = \text{Sky} + \frac{I_c R_c^2}{(R_c^2 + (x - X_c)^2 + (y - Y_c)^2)}$$

to the surface brightness of a globular cluster by minimizing the Mean Absolute Deviation (**MAD**) of the residuals. The simplex algorithm is used to find the minimum. All pixels (except those with **BLANK** values) within a radius **Rmax** from the center are used in calculating the **MAD**. **Sky** : The sky value. **Ic** : The central surface brightness. **Rc** : The core radius. (**Xc,Yc**): The cluster center. –**n nmax** indicates the maximum number of function (**MAD**) calls. –**o image1** indicates that the fitted King model should be saved on **image1**.

22. **line -w fwhm image X1 Y1 X2 Y2**
Prints a table of equally (1 pixel unit) spaced pixels values along a streight line connecting (**X1,Y1**) and (**X2,Y2**). The pixel values perpendicular to the line are weighted by a Gaussian of ajustable width (default width: **FWHM** = 5 pixels). –**w fwhm** specifies the **FWHM** of the Gaussian weighting function. The table is directed to the standard output.

23. **mean -h -s image [Xmin] [Xmax] [Ymin] [Ymax]**
Calculates mean and variance for (**Xmin:Xmax, Ymin:Ymax**). The –**s** option limits the output to a pure mean that can be used as input to other programs. If –**h** is given, a table of bit frequencies is listed, starting with the least significant bit (only type 2).

24. **median -l limit -w file image image1 [image2] [image3]...**
Finds the median or the trimmed weighted mean of a series of images: **image1, image2**, **image3**, ..., and stores the result in the first image. You may use wild card *(s) in **image1**. This program does not find the straight median, if the –**l** option is given. Instead it uses the following procedure: (1) **First**, calculate the Median. (2) **Next**, derive the Median Absolute Deviation (**MAD**) from the median. (3) **Finally**, calculate the weighted Mean of values deviating less than **limit*MAD** from the median. The option –**w file** indicates that the individual weights are read from the text file **file**. If –**w** is not given unit weights are used to calculate the mean. Default is the straight median.

25. **wshow -ipt -r rot -e epdn -w fwhm -f fr -s sr image [Xmin] [Xmax] [Ymin] [Ymax]**
A new show program that can display any subarray **(Xmin:Xmax, Ymin:Ymax)** of any image type (1 – 5). It has the same options (see below) as **show**. In addition it has a few new character modes: **f:** 2-D Gaussian fit, **x:** 1-D Gaussian fit along the X-axis, and **y:** 1-D Gaussian fit along the Y-axis. The starting location is the current cursor position. After the fit **wshow** automatically enters "Move Cursor" mode. The following options, which are of importance for Gaussian fitting, have been added: –**i:** Show imaginary part of complex (type = 5) image. –**e epdn:** Electrons per Digital Number. –**w fwhm:** Guess of Full Width at Half Maximum. –**f fr:** Radius of fitting area. –**s sr:** Inner radius of

sky annulus. The outer radius of the sky annulus is 3***sr**. The following options are of importance for display of the pixel position in relative guide probe coordinates: **–p:** Show relative guide coordinates for P8603 CCD. **–t:** Show relative guide coordinates for TK512 CCD. **–r rot:** Rotator position in degrees. It should not be necessary to use -p or -t for LDS images.

26. **otf -g Gamma -p Power image Xc Yc RadMax NoiseMin NoiseMax**
A program that fits an **Optical Transfer Function** of the form

$$T(u,v) = aH(\omega)\exp(-2\pi i\theta)$$

$$\theta = \alpha u + \beta v$$
$$H(\omega) = \exp(-(\omega/\gamma)^P)$$
$$\omega^2 = u^2 + v^2 + a_2(u^2 - v^2) + 2b_2uv$$

to the complex Fourier transform of a **PSF** star. The zero frequency should have been shifted to the pixel location (**Xc**,**Yc**). All pixels within a radial frequency of **RadMax** Fourier-domain pixels of the zero frequency are included in the fit with unit weight, but the zero frequency is not included, as it represents the sky value. Parcevals theorem ensures that the sum of squares has the same value in the two domains. (α, β) gives the exact location of the **PSF** star in the image domain. Note, that the **PSF** should be shifted to **(0, 0)** before the Fourier transform is taken. The center of the **PSF** should be located within a fraction of a pixel from the origin. γ gives the width of the **OTF** in inverse spatial pixel units. ($\mathbf{a_2}$, $\mathbf{b_2}$) indicates the size of any possible ellipticity, as well as the direction of the ellipticity. The noise level is is estimated from the complex data within the annulus located between the radial frequencies **NoiseMin** and **NoiseMax**. The program lists γ, P, $\mathbf{a_2}$, and $\mathbf{b_2}$ together with standard deviations obtained from the least squares fit.

27. **power image1 image2**
Calculates the power of **image1**. The result is placed in a new float **image2**. The input image can be of any type (1 – 5).

28. **profile -a -s -g -p pa -e eps -n ord -f file -o im1 im Xc Yc Rmin Rmax [npt]**
Calculates radial harmonic profiles for a galaxy by means of a robust regression technique that fits Fourier series to elliptical rings centred on the object. As an option it reconstructs a smooth image between the ellipses given by [**Rmin, Rmax**], where r = sqrt(b*a) is the effective radius. The interpolation is done on the logarithm of the mean intensity profile, **ln(a0[])**, as a function of $r^{\frac{1}{4}}$, if the profile is a monotonic decreasing function, and **a0[npt]** >= 1, otherwise the interpolation is done on the linear profile **a0**[] directly. **–a:** Akima Interpolation. Default is a Cubic Spline. **–s:** Ordinary Least Squares Solution, only. **–g:** A radial gradient is not included in the fit. **–p pa:** Position angle (in deg) of the major axis. **–e eps:** Ellipticity (1 - b/a) of the ellipses. **–n ord:** Order of the harmonic series (max 10). **–f file:** Output file containing a listing of the profiles. **–o im1:** Output image containing the reconstructed image. **npt:** The number of radial points (min 7 rings).

29. **prophot -w width image Xc Yc RadMax [a2] [b2]**
Calculates a radial profile of average pixel values within elliptical rings centred on (**Xc**, **Yc**). –**w width** is used to assign the width of the rings. The default width is 1 pixel. The result is given as a table: Column #1 gives the mean radius of pixels within the ring, column #2 gives the mean pixel value, and column #3 gives the number of pixels within each ring. ($\mathbf{a_2}$, $\mathbf{b_2}$) are optional parameters that defines the ellipticity of the **Optical Transfer Function** as given by the **otf** program.

30. **putkey image**
(test program which assigns all FITS keywords)

31. **rddir directory**
Reads header files back from a single directory file containing many headers. **directory must** be written by wrdir.

32. **rdfits -n -i -s -e ext {image1|@file} [image2] [image3]...**
Translates a (list of) "simple" **FITS** files, **image.ext**, into internal image formats. If the first parameter begins with **'@'** the rest of that parameter is assumed to be the name of a text file containing a list of **FITS** files (**image.ext**) to be converted. If the first parameter does not begin with **'@'** all parameters on the command line are assumed to be names of **FITS** files, without the extension **ext**. The resulting IMSYS image name is the **FITS** file name, minus the extension or, if the –**i** option is given, the name indicated by the FITS keywords **IMAGE** or **FILE**, or if none of these are given the **FITS** file name. The –**e ext** option indicates that **ext** is used as the FITS extension. The **default** extension is "fit". If –**n** is given no word or byte reordering is done while reading the **FITS** files. The **IMAGE** values are derived from **FITS** values by the formula **IMAGE = BSCALE * FITS + BZERO**. The resulting image has the same data type as the FITS file, except when the FITS file is 32-bit integer and **BSCALE** is different from 1.0. Then the resulting image is 32-bit float. The –**s** option indicates that **BSCALE** and **BZERO** should not be used.

33. **rdframe image [npix] [nlin] [frame] [page]**
Reads an image from the Brorfelde frame buffer (386-PC).

34. **rfft -b FltSub image1 image2**
Performs a Fast Fourier Transform of a real image. Because of the symmetry only half of the complex transform is stored. If input **image1** is complex (type = 5), the inverse **FFT** is stored in the float (type = 3) output **image2**. If input **image1** is **not** complex, the forward **FFT** is stored in the complex output **image2**. The real image must be square, and it must have a dimension which is a power of 2. In case **image1** is **not** complex, it must have dimension (N × N), and **image2** will be complex with dimension ((N/2+1) × N). In case **image1** is complex, it must have dimension ((N/2+1) × N), and **image2** will be float with dimension (N × N). –**b FltSub** gives a value which will replace **BLANK** pixels.

35. **runmed -x mx -y my -l limit image image1**
Finds the running median (or trimmed mean) for **image**, and stores the result in **image1**.

The median/mean is calculated for a box of dimension **mx** X **my**. This program does not find the straight median, if the **–l** option is given. Instead it uses the following procedure: (1) **First**, calculate the Median. (2) **Next**, derive the Median Absolute Deviation (**MAD**) from the median. (3) **Finally**, calculate the Mean of values deviating less than **limit*MAD** from the median. Default is the straight median.

36. **setring -s std -p pa -e eps image Xc Yc Rmin Rmax** [**value**]
Writes **value** into a (soft) elliptical ring in an existing image. (**Xc**, **Yc**) is the center of the ring. **image** = **value** for **Rmin** $\leq$ radius $\leq$ **Rmax**. If **–e std** is given, the edge is softened by a Gaussian fall-off to zero with standard deviation **std**. The soft edge is 5 standard deviations wide. **–p pa** gives the position angle (in deg) of the major axis. **–e eps** gives the ellipticity (1 - b/a) of the ellipses.

37. **sclean -l loops -g gain resim psfim clnim**
Finds the highest (or the lowest) pixel value in **resim**. At that location a scaled **PSF**, as given in **psfim**, is subtracted from **resim**, and a value equal to the volume under the scaled **PSF** is added to a single pixel in **clnim**. The unresolved stellar background must have been subtracted. **resim** is the resulting residual image. **–l loops** gives the number of iterations. **–g gain** gives the loop gain (scaling of PSF).

38. **setwin image** [**value**] [**Xmin**] [**Xmax**] [**Ymin**] [**Ymax**]
Writes a value into a rectangle in an existing image.

39. **show image** [**Xmin**] [**Xmax**] [**Ymin**] [**Ymax**]
Image display program for 386-PC with either the **Paradise** VGA board, or any board based on the Tseng Labs **ET4000** Graphics Controller. It displays any subarray **(Xmin:Xmax, Ymin:Ymax)** of type = 2 images (16-bit integer). A few specific input characters are used to enter various modes which allow you to change cursor colour, to move the cursor etc. by means of the keyboard arrows, and special keys. The show program has the following modes:

Character Mode	Function
c	Change Cursor Colour
s	Change Cursor Step Size(s)
h	Change Starting Hue for Lookup Table
g	Use Gray Scale in "Move Cursor" mode
m	Move Cursor with given Step Size(s)
e	Exit Show Program

Values are changed and the cursor is moved by pressing the keyboard arrows. A new mode is entered by pressing one of the above characters. The m-mode allows two speeds. The slow speed is controlled by the regular arrows, and the fast speed is controlled by <Delete>, <Page Down>, <Home>, and <End>. The speed is changed by entering the s-mode. The two step sizes are changed by using the regular arrows, or the special keys mentioned above, respectively.

40. **spot image [X] [Y] [Imag]**
    Imag = { 0 | 1 } (selects real or imaginary part, if complex)
    Writes an 11×11 square centred on **(X, Y)** onto the terminal.

41. **wrdir directory**
    Stores all header files onto a single ASCII directory file.

42. **wrfits -s scale -z zero image1 [image2] [image3]...**
    Translates an IMSYS "simple" image into a file **image.fit** in **FITS** format. You may use wild card *(s) in image1. But remember: wrfits "ngc*" on Unix systems. **wrfits** introduces a non-standard **FITS** keyword **IMAGE**, which indicates the original IMSYS name. −**s scale** gives the FITS **BSCALE** value used (default = 1). −**z zero** gives the FITS **BZERO** value used (default = 0). The **FITS** values are derived from **IMAGE** values by the formula **FITS = (IMAGE − BZERO)/BSCALE**. The resulting **FITS** file has the same data type as the IMSYS image, except when the image type is 32-bit float and **BSCALE** is different from 1.0. Then the resulting FITS file is 32-bit integer. BSCALE is only used for 32-bit float images. In addition, BZERO is also used for 16-bit and 32-bit integer images.

43. **wrframe image [frame] [page]**
    Writes an image into the Brorfelde frame buffer (386-PC).

44. **xtract -u image1 image2 [type] [Xmin] [Xmax] [Ymin] [Ymax]**
    Extracts a new image (**image2**) of indicated **type** from
    an existing image (**image1**) of any type. If −**u** is given, the input picture **image1** is treated as unsigned, if it is of type short int (type 2).

IMSYS is currently running on the following compilers:

a. Microsoft-C V5.1 + MS-DOS V3.3

b. Zortech-C V2.1 + MS-DOS V3.3

c. Microsoft-C/C++ V7.0 + MS-DOS V5.0

d. High-C R1.6 + Phar Lap's 80386—DOS-Extender

e. High-C R2.31 + Phar Lap's 80386/486—DOS-Extender

f. High-C R2.33 + Phar Lap's 80386/486—DOS-Extender

g. High-C/C++ VR.04 + Phar Lap's 80386/486—DOS-Extender

h. WATCOM-C/386 V9.0 + Phar Lap's 80386/486—DOS-Extender

i. WATCOM-C/386 V9.0 + Rational Systems DOS/4GW

j. cc under Unix System V (HP-UX V7.05/8.05/9.01) on HP-380/835/710

k. cc under BSD Unix (CONCENTRIX) on an Alliant FX8

l. cc under BSD Unix (ULTRIX V4.1) on a RISC DecStation 3100

m. cc under BSD Unix (SunOS V4.1.3) on a Sun SparcStation II

n. cc under Unix V.3 (IRIX V4.0.2) on a SiliconGraphics IRIS Indigo

o. cc under Unix (ConvexOS V10.0) on Convex 3220 and Convex 3830

p. gcc 2.7.2 under Linux 2.0.27 on a Pentium PC with a PCI-bus

q. gcc 2.6.2 under FreeBSD 2.0 on a Pentium PC with a PCI-bus

The compiler/OS combination is selected by using the proper include file include/host.h. There are 10 versions of host.h, and 5 of ftnif.h:

A. host.msc (Microsoft-C and Zortech-C)

B. host.hc1 (High-C Version 1.6)

C. host.hc2 (High-C Version 2.31, 2.33, and 3.04)

D. host.wc (WATCOM-C/386 V10.0 under Rational Systems DOS/4GW)

E. host.wcp (WATCOM-C/386 V9.0 under Phar Lap's DOS-Extender)

F. host.bsd (Concentrix cc Alliant; no setvbuf() function)

G. host.dec (ULTRIX V4.1 cc and SunOS V4.1.3 cc)

H. host.sig (IRIX V4.0.2, HP-UX V9.01, Linux 1.1.59, FreeBSD 2.0)

I. host.cvx (ConvexOS V10.0 cc on Convex 3220 and Convex 3830)

J. host.v (HP-UX V7.05/8.05 cc on HP-380, HP-835, and HP-710)

K. ftnif.300 (Fortran Interface on HP-380)

L. ftnif.800 (Fortran Interface on HP-835)

M. ftnif.dec (Dec 3100, SparcStation II, IRIS Indigo, Convex 3220)

N. ftnif.wat (486-PC + WATCOM-C/386 V10.0 + Rational Systems DOS/4GW)

O. ftnif.f2c (Pentium PC + f2c + gcc + Linux 2.0.27 or FreeBSD 2.0)

One of host.* should be copied onto host.h, and one of ftnif.* should be copied onto ftnif.h

The main directory contains the following subdirectories:

1. include: Include files
2. libsrc: Library source files

3. utilsrc: Utility programs
4. ftnsrc: Fortran test program
5. obj: Object files for IMSYS
6. lib: Library files
7. bin: Executable files
8. bat: DOS batch files
9. make: Make files
10. etc: Some batch files, text files, and readme.doc
11. lharc: The LHarc packing program

The whole system is compiled and linked by using make on 1 of 19 make files, depending on the operating system or compiler version:

1. make imsys (MS-C without floating point adapter/MS-make)
2. make imsysfp (MS-C with floating point adapter/MS-make)
3. make -fimsyszt (Zortech-C with floting point adapter/ZT-make)
4. nmake -f imsysc7 (MS-C/C++ V7.0/386-PC/MS-nmake)
5. make imsyshc1 (High-C R1.6/386-PC/MS-make/PharLap run386)
6. make imsyshc2 (High-C R2.31/386-PC/MS-make/PharLap run386)
7. make imsyshc3 (High-C R2.33/386-PC/MS-make/PharLap run386)
8. make imsyshc4 (High-C R3.04/386-PC/MS-make/PharLap 386stub.exe)
9. mwmake -f imsysmw (High-C R3.04/486-PC/MW-make/PharLap 386stub.exe)
10. wmake -f imsyswc (WATCOM-C/386 V9.0/WC-wmake/Rational Sys. DOS/4G)
11. wmake -f imsyswcp (WATCOM-C/386 V9.0/WC-wmake/PharLap 386stub.exe)
12. make -f imsys5 (Unix System V: HP-UX V7.05/8.05/9.01/Unix-make)
13. make -f imsys700 (Unix System V: HP-UX V9.01/ANSI-Ext/Unix-make)
14. make -f imsysb (BSD Unix: Concentrix/Unix-make)
15. make -f imsysdec (BSD Unix: ULTRIX V4.1 and SunOS V4.1.3)
16. make -f imsyssig (Unix System V.3: IRIX V4.0.2/Unix-make)
17. make -f imsyscvx (Unix: ConvexOS V10.0/Unix-make)

18. make -f imsyslnx (Unix: Linux 2.0.27)

19. make -f imsysbsd (Unix: FreeBSD 2.0)

Make should be called from directory imsys/make

MetaWare's High-C R1.6, R2.3, and R3.0 are sufficiently different that they require different makefiles. PharLap's stub-loader has now been linked with all programs in order to avoid using run386 explicitly. MetaWare's mwmake has been introduced in order to drop MicroSoft's utilities in the future. WATCOM's C9.0/386 compiler with its own wmake utility has been introduced due to errors in High-C R3.0. NOTE that show, wshow, rdframe, and wrframe only work when they are compiled by MS-C, High-C, or WATCOM-C/386.

On UNIX 4 environment variables MUST be defined in the .login file:
setenv HDRPATH <path to directory containing header files>
setenv IMGPATH <path to directory containing image files>
setenv ARGS <path to directory containing argument files>
setenv IMSYS <path to the IMSYS root directory>
In addition, if IDL is implemented:
setenv IDL_PATH .:$IMSYS/etc:.....

On MS-DOS the 4 environtment variables MUST be defined in the autoexec.bat file:
set HDRPATH = <path to directory containing header files>
set IMGPATH = <path to directory containing image files>
set ARGS = <path to directory containing argument files>
set IMSYS = <path to the IMSYS root directory>

IMSYS can handle 5 different image types:

1. unsigned char (type = 1) 8bit unsigned integer

2. short (type = 2) 16bit signed integer

3. float (type = 3) 32bit real

4. long (type = 4) 32bit signed integer

5. complex (type = 5) 2x32bit reals

There is no **show** program under Unix, but there is an interface package imsys.pro in directory imsys/etc which can be used to read and write IMSYS images from within IDL (Interactive Data Language, Research Systems, Inc., Boulder, CO 80303). Information on these procedures may be obtained by the IDL call:

DOC_LIBRARY, 'imsys'

The following user procedures are implemented: 'getim', 'putim', 'newim', 'irm', and 'list'. I have also modified the X11 display program **SAOImage** so that it can read IMSYS images of types **uchar, short, float**, and **long**.

IMSYS has a library module containing functions, which can be called from Fortran. There is a small test program called test.f in imsys/ftnsrc. The calling procedure for the interface functions is described in the source file: imsys/libsrc/ftnif.c The trailing underscore must be omitted on the Alliant. Unfortunately, the convention for passing strings depends on the operating system. It is defined in ftnif.h. Remember that all text strings **must** be terminated by char(0) in C. A new function igenv_() has been added to ftnif.c. It allows a Fortran program (e.g. **DAOPHOT**) to access the standard C function getenv().

Installation procedure on a DEC or SUN workstation:

1. mkdir <directory containing header files>
2. mkdir <directory containing image files>
3. mkdir imsys
4. cd imsys
5. tar xvf /tmp/imsys.tar
6. cd make
7. make -f imsysdec
8. cd ..
9. define the environment variables HDRPATH and IMGPATH in the login file. (have a look at the login file in imsys) add imsys/bin to the PATH string in the login file

The system should now be ready to use...

At the Institute of Physics & Astronomy, Aarhus, the old version of **DAOPHOT** has been replaced by Stetson's newer version **DAOPHOT-II** (dated November 1991). It is also using the IMSYS interface, but it is located in its own directory. **DAOPHOT-II** is running on the HP-300 and HP-800 series computers, as well as on the RISC Decstation 3100 and on a 486-PC using the WATCOM F77/386 V10.0 compiler. **DAOPHOT-II** has also been compiled and linked under Linux 2.0.27 and FreeBSD 2.0. Images can be inspected from within IDL by using the IDL interface, or by using the **SAOImage** display program.

It is possible to use wild card (*)s in **wrfits**, **list**, and **irm**. It is also possible to read all FITS files with extension ".fit" by the commands: ls *.fit > lst followed by rdfits @lst

It is now possible to fit both 2-D, as well as 1-D, Gaussian functions to stellar or slit images near the current cursor position within a new display program **wshow**. It can display images of any type (1 - 5). The stand-alone program **gauss** can still be used. It is also possible to fit a 1-D Gaussian to a slit image by selecting '-x' or '-y' as the optional parameter for gauss. The

2-D Gaussian takes the finite pixel size into account.

The current version is compatible with Microsoft Windows 3.1.

Bjarne Thomsen
Institute of Physics and Astronomy
Ny Munkegade
DK-8000 Aarhus C
Denmark
E-mail: bt@obs.aau.dk (Internet)
Fax: 45 86 12 07 40