Software Structure and Its Performance on FOCAS Instrument Control, a MOS Design, and an Analyzing Package

Michitoshi Yoshida[a], Yasuhiro Shimizu[a], Toshiyuki Sasaki[b], George Kosugi[b], Tadafumi Takata[b], Kaz Sekiguchi[b], Nobunari Kashikawa[c], Kentaro Aoki[d], Ryo Asai[e],Youichi Ohyama[c], Koji Kawabata[c], Motoko Inata[c], Yoshihiko Saito[f], Hiroko Taguchi[g], Noboru Ebizuka[h], Yasushi Yadoumaru[i], Tomohiko Ozawa[i] and Masanori Iye[c]

[a]Okayama Astrophysical Observatory, National Astronomical Observatory of Japan, Kamogata, Okayama, 719-0232 Japan
[b]Subaru Telescope, National Astronomical Observatory of Japan, 650 North A'ohoku Place, Hilo, Hawaii, 96720 USA
[c]Optical and Infrared Astronomy Division, National Astronomical Observatory of Japan, 2-21-1 Osawa, Mitaka, Tokyo, 181-8588 Japan
[d]Astronomical Data Analysis Center, National Astronomical Observatory of Japan, 2-21-1 Osawa, Mitaka, Tokyo, 181-8588 Japan
[e]Systems Engineering Consultants, Co., LTD., Shibuya-ku, Tokyo, 150-0031 Japan
[f]Department of Astronomy, The University of Tokyo, 7-3-1 Hongo, Bunkyo-ku, Tokyo, 113-8654 Japan
[g]Department of Astronomy and Earth Sciences, Tokyo Gakugei University, Koganei, Tokyo, 184-8501 Japan
[h]Communication Research Laboratory, Koganei, Tokyo, 184-8795 Japan
[i]Misato Observatory, 180 Matsugamine, Misato, Wakayama, 640-1366 Japan

ABSTRACT

Faint Object Camera And Spectrograph (FOCAS) is completed and now waiting for a commissioning run on the Subaru Telescope atop Mauna Kea. We have developed a software system that includes the control of FOCAS instruments, Multiple Object Slits (MOS) design, and an analyzing package especially for evaluating performances of FOCAS. The control software system consists of several processes: a network interface process, user interface process, a central control engine process, a command dispatcher process, local control units, and a data acquisition system. These processes are mutually controlled by passing messages of commands and their status each other. The control system is also connected to Subaru Observation Software System to achieve high efficiency and reliability of observations. We have two off-line systems: a MOS design program, MDP, and an analyzing package. The MDP is a utility software to select spectroscopy targets in the field of view of FOCAS easily through its GUI and to design MOS plates efficiently. The designed MOS parameters are sent to a laser cutter to make a desirable MOS plate. A special package enables prompt performance check and evaluation of the FOCAS itself during a commissioning period. We describe the overall structure of FOCAS software with some GUI samples.

Keywords: Instrumentation, Subaru telescope, Spectrograph, Software design, Control software, Analyzing software, MOS design

1. INTRODUCTION

FOCAS[1] is an optical camera and spectrograph to be attached at the Cassegrain focus of Subaru telescope.[2] The main purpose of this instrument is to enable deep optical spectroscopy of faint objects: i.e. distant galaxies, quasars, faint envelope of galaxies and gas nebulae, outer asteroids of Solar system, etc. FOCAS has three filter turrets, a

Send correspondence to M. Yoshida. E-mail: yoshida@optik.mtk.nao.ac.jp

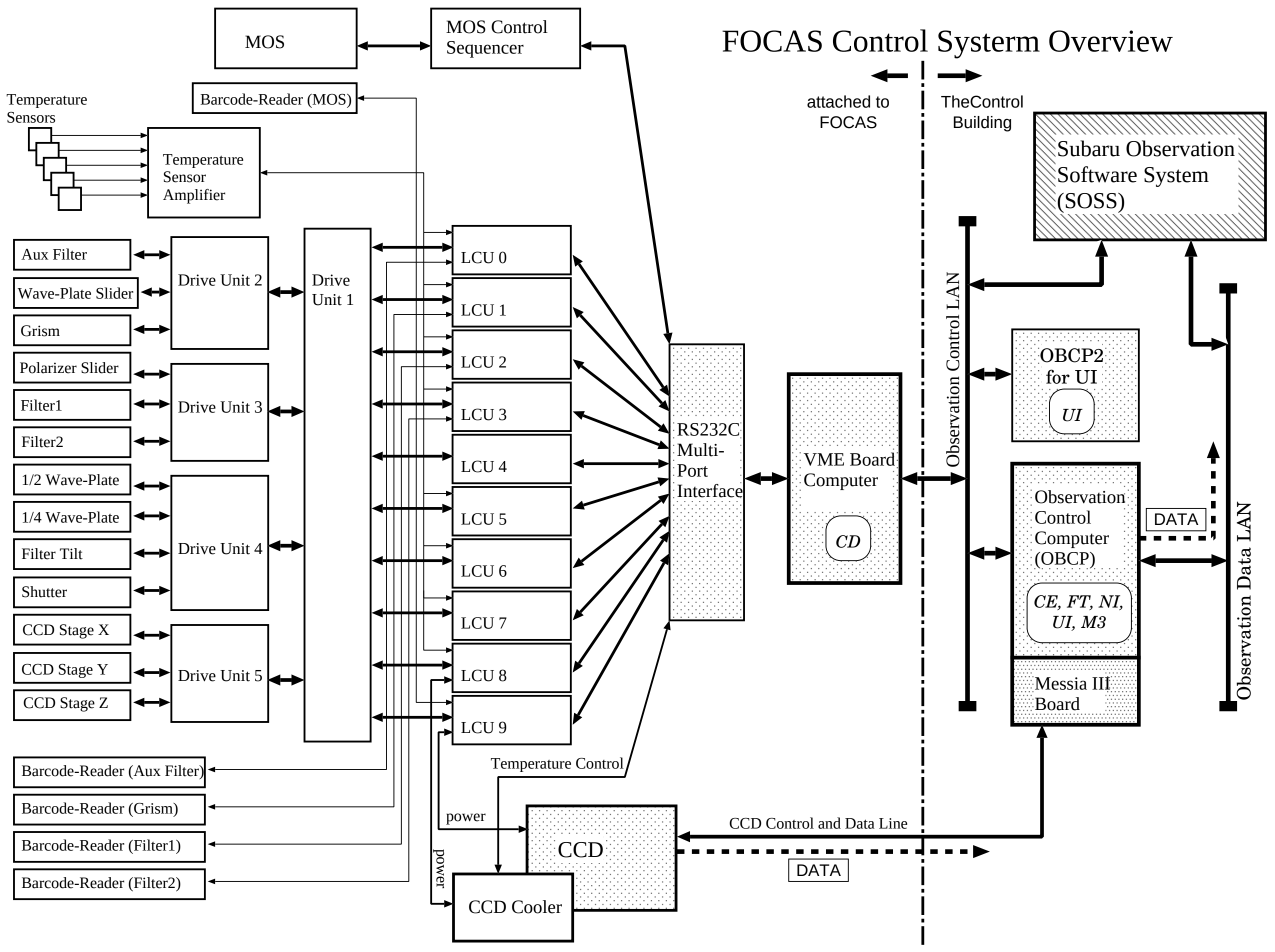


Figure 1. The FOCAS control system overview. This figure presents mainly the hardware configuration of the system. The components shown in left-hand side of the dot-dash line are attached to the instrument, whereas the components in right-hand side are installed in the control building of Subaru telescope. CCD image data are sent to the OBCP from the CCD controller through a high-speed optical fiber channel.

grism turret, two sliders, two rotators for waveplates rotation, a XYZ stage of the detector mount, and a shutter in its main beam. One of the filter turrets has a filter-tilting mechanism for adjust filter transmission curve to shorter wavelength. The multi-object slits (MOS) plates exchange system is installed at the Cassegrain focal plane of the telescope. The detector is a mosaic CCD which consists of two 2048 $\times$ 4096 pixel SITe CCD chips.

We developed not only an instrument control software of FOCAS, but also two offline systems: one of which is a MOS design software and the other is a data analysis package. The MOS design software helps observer to select spectroscopy targets from imaging data and design MOS plates. The outputs from the software are sent to a laser cutting machine. The data analysis package is optimized for evaluating performances of FOCAS. We describe below the above three software systems: the instrument control software, MOS design software, and the data analysis package.

2. FOCAS CONTROL SOFTWARE FALCON

2.1. Overview of the FOCAS Control System

The configuration of the FOCAS control system[3] is shown in Figure 1. The motor drive units directly drives the motors of the movable devices of FOCAS and receives the outputs from the encoders of the devices. Local control

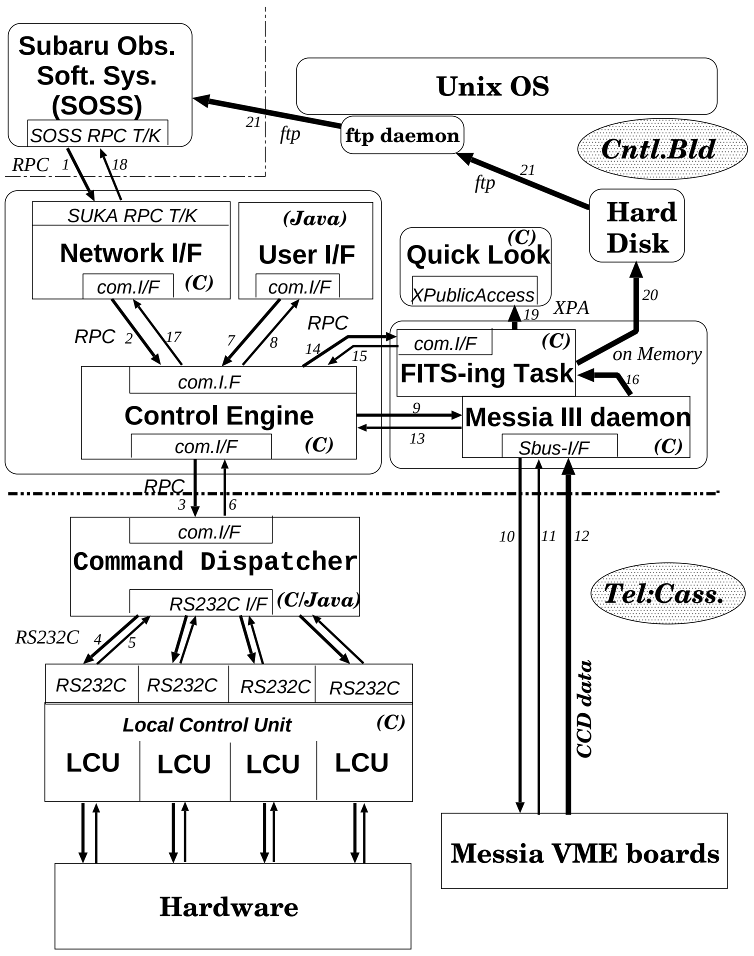


Figure 2. FALCON internal structure. Commands input from UI or SOSS through NI are transmitted to CE, and CE checks the commands and dispatch them to appropriate processes, CD, M3, or UI.

units (LCUs), which are one-chip micro-computer boards, control the movable devices through the motor drive units. LCUs control also the barcode readers installed at the filter turrets and the powers of the CCDs, the cooler and the insulator valve. A VME board computer which is installed in a thermal insulation box attached at FOCAS is the host computer of LCUs. All the LCUs are asynchronously controlled by the host computer through a multi-port RS232C interface. The VME board computer communicates also with the sequencer of the MOS exchange system through one port of the multi-port RS232C, and controls the MOS system. The observation control computer (OBCP) of FOCAS is in the control building of the Subaru telescope and the main part of the control software of FOCAS runs on this machine. The CCDs are directly controlled by the OBCP with the Messia III system. The CCD data is transferred to the OBCP through a high-speed optical fiber channel and are stored in a shared memory of the OBCP. Whole system is network-connected to the Subaru Observation Software System (SOSS)[4].[5]

2.2. Conceptual Design of FALCON

The instrument control software of FOCAS (FALCON; Focas ALlround CONtrol software)[3] is designed as a network distributed, multi process system. It consists of several internal processes: a network interface process, user interface process, a central control engine process, a command dispatcher process, local control units, and a data acquisition system. The data acquisition system is composed of a Messia III daemon process and a data FITSing task. One of the advantages of this type of thin connected, multi-process system is ease of maintenance of the system; i.e. it is very to debug a process without stopping the whole system. The structure of FALCON is shown in Figure 2. Each process is independently runnable and communicates each other by passing command strings using RPC (Remote Procedure Call). We describe briefly the format of the command strings in the following section.

2.3. FALCON Internal Commands and Control Objects

A FALCON internal command (hereafter FALCON-command) used in communication between the internal processes of FALCON is composed of a fixed length (69 bytes) header and a command sentence. The header contains total length of command, date and time, sequential number of command, name of command transmitting process, and name of command destination process. Command sentence is variable in length. A command sentence is composed of verb, destination process/instrument, command object, control objects, and parameters of the control objects. The verbs defined are "exec", "require", "send", and "set". The verb "exec" is mainly used to drive the devices of FOCAS which are indicated as the control objects. For example, the following command sentence drives one of the filter turrets ("Filter1") to a particular position ("1"):

```
exec FOCAS object Filter1.Motor=ON Filter1.Select=1 .
```

The verb "require" is to read the status of devices and "send" is to send the status of the devices to other processes. The protocol of FALCON command sentence is summarized in Table 1.

Table 1. FALCON command table.

Control Class	Commands
	Execution Command
Execution Command	`exec PID`[†] `Object` *Parameters* (The format of *Parameters* is as follows: *object.argument=value*)
	System Control Commands
Emergency Stop	`exec PID Emergency_stop`
Initialization	`exec PID Initialize`
Process Termination	`exec PID Exit`
Process Diagnostics	`exec PID Diagnose`
Authentication	`exec FOCAS Authenticate User=username Mode=[Operation|Maintenance]`
	Status Handling
Status Transmission Mode Setting	`set CE Status Keyword=[KeywordList|All|Default] Cycle=[0|n|EVENT]`
Status Request	`require PID Status` *Parameters* (The format of *Parameters* is as follows: *object.argument*)
Status Transmission	`send PID Status` *Parameters* (The format of *Parameters* is the same as Execution Command)
	Control Priority Manipulation
Control Priority Request	`require PID Control Priority=[High|Low|None]`
Priority Manipulation	`exec PID Control Priority=[High|Low|None]`
	Mode Setting
Operation Mode Setting	`set PID Mode Operation=[Simulation|Operation|Maintenance]`
Control Mode Setting	`set PID Mode Control=[Local|Remote]`
Environment Setting	`set FOCAS Mode Environment=[Telescope|Dome|OptSim|RD]`
	Data Acquisition
Frame Number Acquisition	`exec FOCAS FRAME_NO Mode=Get`
Frame Number Transmission	`exec FOCAS FRAME_NO Mode=Send Frame=`*FrameNo.*
Data Transmission	`exec FOCAS OBS_DATA Mode=Transfer Frame=`*FrameNo.*

† Object process ID. "FOCAS", "CD", "M3", "FT", "UI", "NI" or "CE' can be set as a PID.

2.4. Individual Process

2.4.1. Local Control Unit (LCU)

LCU is a custom-made one-chip microcomputer (H8/3048F-ZTAT) board which has 8 channel DC motor drivers, 11 I/O ports (61 bit available), two analog I/O ports and two RS232C interfaces. One of the RS232C is used to communicate a host computer. LCU built-in software is installed into the rewritable ROM of the CPU. The software provides a simple command protocol (hereafter LCU-command) which is used for communication between LCUs and host computer. All the movable elements of FOCAS except for the MOS system can be monitored and controlled by sending LCU-commands to LCUs through RS232C. FOCAS has 10 LCUs. Each of those controls a few devices simultaneously, and monitors the temperature sensors which are installed at various positions of FOCAS.

2.4.2. Command Dispatcher (CD)

CD lies above LCU in the FALCON hierarchy. It is a Java program and runs on the VME board computer. CD receives FALCON-commands from CE, analyses them and translates them into LCU-commands, then dispatches the commands to appropriate LCUs. It also maintains execution status of received FALCON-commands and status of the devices of FOCAS. In order to implement asynchronous, parallel execution of commands, we adopted Java multi-thread API as the fundamental structure of CD. The control devices of FOCAS are implemented as objects (Java classes) each of which has translation mechanism between FALCON-commands and LCU-commands and maintains the command status and the device status. CD has an internal command dispatcher thread, a status management thread, a command execution watcher, network interface threads, and RS232C interface threads. Those threads run in parallel and communicate each other through internal command buffers. Because the standard Java API does not support RS232C communication, we developed new classes which invoke native codes written in C using the Java Native Interface (JNI) API. We use the "ONC RPC/XDR Toolkit for Java" which is a product of Distinct Corporation[*] for implementation of the RPC client and the RPC server of CD.

2.4.3. Messia III Daemon (M3)

M3 interfaces the Messia control software[6] with CE. The Messia control software controls the general-purpose CCD control system, Messia III. M3 adds a RPC interface at the command input gate of the Messia software and receives the responses from the Messia through pipe. In addition, M3 controls the CCD exposure process; i.e. shutter control, wiping CCD (readout residual electrons from CCD), and CCD readout.

2.4.4. Fitsing Task (FT)

After finishing a CCD readout, the readout data are stored in a shared memory of the OBCP. FT reads the data in the shared memory and creates a FITS file with the telescope and instrument status. The telescope status is retrieved from SOSS through the control engine process (CE) and the network interface process (NI). The instrument status is obtained from CD through CE. FT constructs FITS header from the status information and saves the created FITS file to the local disk of the OBCP. Then FT informs SOSS of the location of the FITS file through CE and NI. Then SOSS creates a FTP link to the OBCP and transmits the indicated FITS file to the Subaru data archive system.[7]

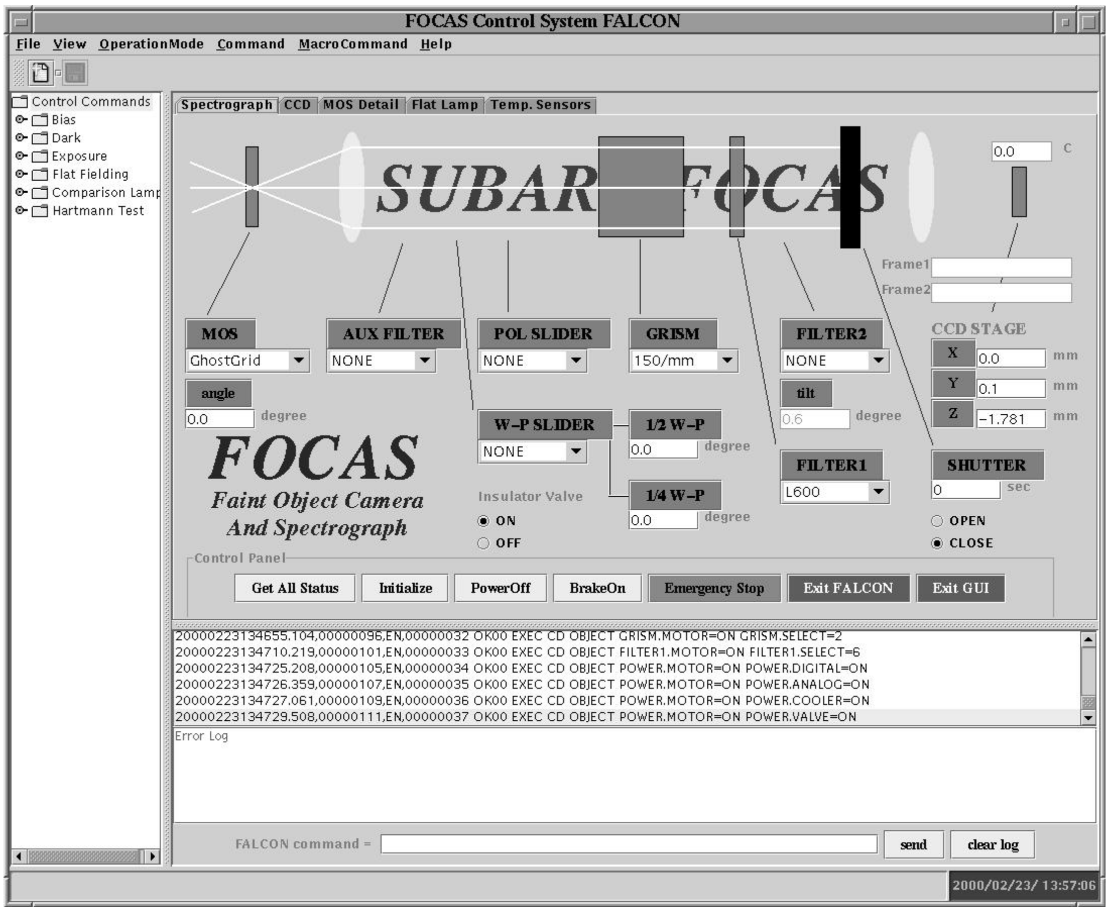
2.4.5. User Interface (UI)

UI is written in Java (Figure 3). Java Swing API is used to construct the graphical interface. Since the source code of UI is pure Java, the compiled byte code of UI can be run on any platforms which support JDK 1.1 or later. UI displays all the instrument status and the command execution status notified by CE. All the devices of FOCAS can be controlled from UI. UI also communicates with FALCON by RPC ("ONC RPC/XDR Toolkit for Java" is used). This thin connectivity to the main engine of the software, together with the platform independent characteristics of Java, makes UI significantly portable. Since UI is not deeply built in the system, it is quite easy to customize its graphical layout or even to develop a thoroughly new UI.

2.4.6. Network Interface (NI)

NI interfaces FALCON with the Subaru Observation Software System (SOSS).[8] NI consists of the SOSS interface toolkit and a translator between SOSS commands and FALCON commands. FOCAS is controlled by command input from SOSS in its normal operation mode. NI receives the instrument dependent commands from SOSS and translate them into FALCON commands, then sends them to the Control Engine (CE) described below.

[*]http://www.distinct.com



```
MOS: GhostGrid
angle: 0.0 degree
AUX FILTER: NONE
POL SLIDER: NONE
GRISM: 150/mm
FILTER2: NONE
tilt: 0.6 degree
W-P SLIDER: NONE
1/2 W-P: 0.0 degree
1/4 W-P: 0.0 degree
Insulator Valve: (●) ON  ( ) OFF
FILTER1: L600
0.0 C
Frame1
Frame2
CCD STAGE
X 0.0 mm
Y 0.1 mm
Z -1.781 mm
SHUTTER: 0 sec
( ) OPEN  (●) CLOSE
```

```
20000223134655.104,00000096,EN,00000032 OK00 EXEC CD OBJECT GRISM.MOTOR=ON GRISM.SELECT=2
20000223134710.219,00000101,EN,00000033 OK00 EXEC CD OBJECT FILTER1.MOTOR=ON FILTER1.SELECT=6
20000223134725.208,00000105,EN,00000034 OK00 EXEC CD OBJECT POWER.MOTOR=ON POWER.DIGITAL=ON
20000223134726.359,00000107,EN,00000035 OK00 EXEC CD OBJECT POWER.MOTOR=ON POWER.ANALOG=ON
20000223134727.061,00000109,EN,00000036 OK00 EXEC CD OBJECT POWER.MOTOR=ON POWER.COOLER=ON
20000223134729.508,00000111,EN,00000037 OK00 EXEC CD OBJECT POWER.MOTOR=ON POWER.VALVE=ON
Error Log

FALCON command =
```

Figure 3. The main panel of FALCON User Interface (UI).

2.4.7. Control Engine (CE)

CE supervises and harmonizes all the processes running in FALCON. It makes format checking and grammatical checking of input commands, and dispatches them to appropriate processes. It maintains the running status of each process and the instrument status which is reported by CD. In full functional, normal operation mode, the processes in FALCON communicate each other through CE (see Figure 2).

2.5. Control Sequence – Command Flow in FALCON

The outline of the command flow in FALCON is as follows. A FALCON-command input from UI or SOSS is first transmitted to CE. CE checks the command format and grammatical error, then checks the validity of the control objects and their parameters indicated in the command sentence. After checking the command validity, CE sends the command to appropriate processes. The process receiving the command executes it and returns a completion message to CE when the execution is ended. Then CE notifies UI and SOSS of the command completion.

In normal operation mode, FOCAS is controlled from SOSS. In order to achieve high observation efficiency, "observation abstract commands"[4] can be defined by instrument developer or user in SOSS. An abstract command is a macro command representing an observation sequence. In actual implementation, an abstract command is composed of a sequence of "instrument dependent commands". An instrument dependent command corresponds a control step of the instrument. In SOSS an abstract command is decoded into a sequence of instrument dependent commands, then the instrument dependent commands are sent to the instrument through the SOSS interface toolkit.

As described in the section 2.4.4, CCD data obtained by the Messia III system is transmitted to shared memory of the OBCP by M3, and is formatted in FITS by FT. Afterwards, SOSS transmits the FITS file to the Subaru data archive system.

2.6. Performance of FALCON

FALCON was almost completed in late 1999 and has been tested in test runs at Mitaka campus of NAOJ and at Hilo base of Subaru telescope. We have confirmed that all the fundamental commands including optical elements drive, CCD drive, status handling, and data transfer work well. Communication with SOSS including data transfer to the Subaru archive system also has no problem. Thus, FOCAS can now be controlled using FALCON thoroughly from its local user interface (UI) and SOSS.

The command response time of FALCON is typically an order of 0.1 seconds. It depends mainly on the period of timer interruption of CE. To avoid heavy load on the OBCP, CE manipulates command messages within time interval of 50 msec. Further, overhead on the synchronization of multi threads of Java may affect the response of CD. Although the amount of this overhead cannot be estimated precisely, it should not be over a few tens msec. The build-in software in LCUs also uses a 40 msec timer interruption in itself. Ignoring network delay of RPC, we thus estimate the total command response time of FALCON is at most 100 - 200 msec. This estimate is confirmed in the test runs. This response is sufficiently fast to control FOCAS whose time constant is chiefly determined by the slow readout time – 2 minutes in typical – of its detector.

The shutter control is the most time critical for FOCAS. In FALCON, one LCU is allocated to control the shutter exclusively and make a time control by using internal timer. Because the time control is made in the LCU locally, the accuracy of time of an exposure depends solely on the internal timer of the LCU. As described above, since the maximum time delay of execution of the built-in software in LCU is 40 msec, a time counting error of at most 80 msec may occur in the time control. This value is sufficiently shorter than the shortest drive time (≈ 0.3 sec) of the shutter hardware.

3. MULTI-OBJECT SLIT PLATE DESIGN PROGRAM, MDP

3.1. MOS Data Flow

The procedure of making a multi-slit plate is as follows (see Figure 4). First an image of the target region is obtained in FOCAS imaging mode. The image is then reduced and all the astronomical objects in the image are automatically detected using an automatic object detection software, SExtractor.[9] The original image and the object list file output from SExtractor are input to the multi-object slit plate design program, MDP, to select spectroscopy target objects and to design MOS plates. MDP creates text files in which the positions and the shapes of slits are written in CCD coordinate (in pixel units). Each text file corresponds a MOS plate. Then, a coordinate transformation including image distortion correction is made for the slit positions and shapes to obtain those in the telescope Cassegrain focal plane coordinate (in millimeters). The resultant data are used to make MOS plates with the laser cutting machine, and are also used by FT in FALCON to construct FITS file.

Typically, it takes about 1 - 2 hours to perform the above procedure and make slit position data. The laser cutting machine is able to cut 50 slits each of which have a length of 10 arcsec and a width of 0.5 arcsec in one hour. Thus, the total time to create one MOS plate is about 2 - 3 hours in typical case.

3.2. MDP

MDP is a utility software to make it easy to select several tens of spectroscopy targets in an imaging data and to design MOS plates. MDP is written in Java. MDP reads an imaging data and an object list file output by SExtractor, and display them. The object list file is displayed as a text table and is also overlaid on the imaging data (see Figure 5). User can set several design parameters such as slit length, slit width, object brightness range, object size range, or galaxyness parameter, etc. in selecting spectroscopy targets and determining slit positions and shapes. Two design modes are prepared in MDP: automatic design mode and interactive design mode. In automatic mode, MDP selects targets based of the design parameters set by user automatically. When overlapping of slits is occurred, overlapped slits are layout to other MOS plates. MDP can design at most 10 MOS plates simultaneously from one imaging data. In interactive mode, user can design MOS plates using mouse.

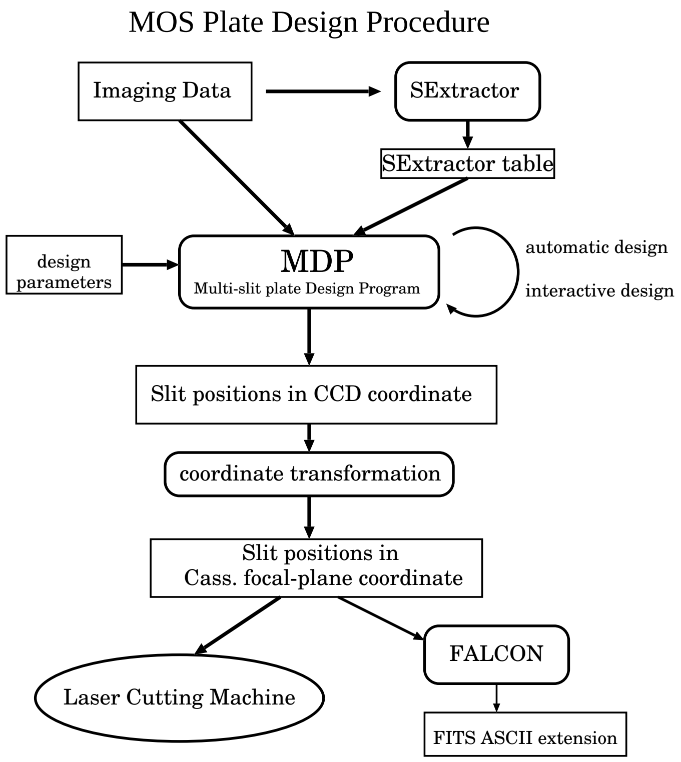
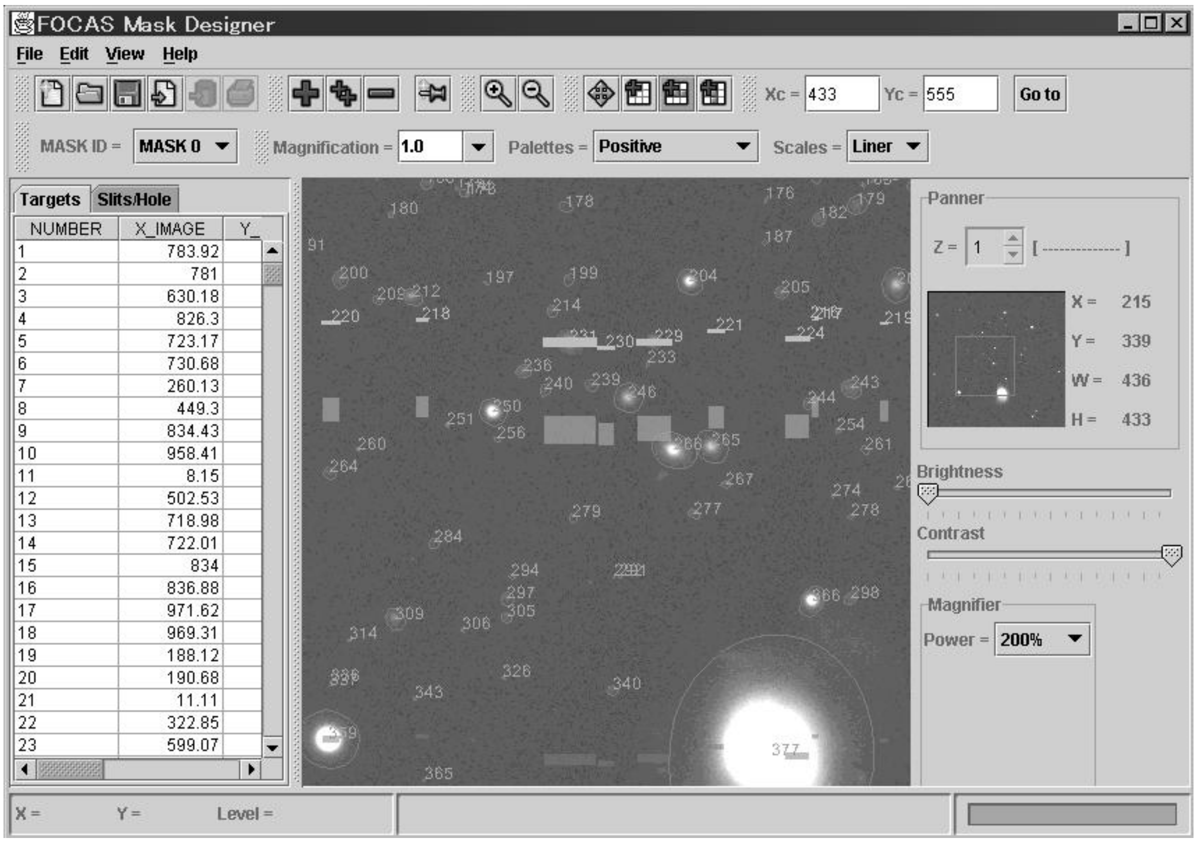

MOS Plate Design Procedure
Imaging Data
SExtractor
SExtractor table
design parameters
MDP
Multi-slit plate Design Program
automatic design
interactive design
Slit positions in CCD coordinate
coordinate transformation
Slit positions in Cass. focal-plane coordinate
Laser Cutting Machine
FALCON
FITS ASCII extension

Figure 4. MOS plate design procedure.

4. DATA REDUCTION PACKAGE FOR SETUP AND PERFORMANCE VERIFICATION OF FOCAS

The data reduction package mainly used for FOCAS performance verification is developed on the image processing software IDL (Figure 6). The package provides several tasks to make a quick reduction of FOCAS data and to feedback to the instrument. The following tasks are currently available:

- Hartmann Test: This task measures automatically the center positions of slit images in two CCD frames and obtains the position difference between the slit images. This is useful to make a Hartmann test in focussing the instrument camera.
- Slit Alignment: This task traces automatically the center position of a long slit image along the direction of the slit length and calculates its tilt angle relative to the CCD row.
- Grism Alignment: Fundamental function of this task is the same as the "Slit Alignment" task, but the tracing direction is perpendicular. This task is used to measure the tilt angle of continuum spectrum image of a point source relative to the CCD column.
- Telescope Focussing: After focussing the instrument camera using Hartmann test, focussing the telescope is made by a multiple exposure technique. This task makes two dimensional Gaussian fitting for star images and obtains the minimum of the FWHMs of the images.
- Image Distortion: In order to measure the image distortion of FOCAS, we put a square grid pattern mask plate on the entrance focal plane and obtain its image. This task measures automatically the positions of the grid pattern projected on the CCDs and fits two-dimensional polynominal function to the data for obtaining the image distortion function of FOCAS (see Figure 6).



Xc = 433 Yc = 555

MASK ID = MASK 0 Magnification = 1.0 Palettes = Positive Scales = Liner

Z = 1 X = 215 Y = 339 W = 436 H = 433

Power = 200%

NUMBER	X_IMAGE	Y_
1	783.92	
2	781	
3	630.18	
4	826.3	
5	723.17	
6	730.68	
7	260.13	
8	449.3	
9	834.43	
10	958.41	
11	8.15	
12	502.53	
13	718.98	
14	722.01	
15	834	
16	836.88	
17	971.62	
18	969.31	
19	188.12	
20	190.68	
21	11.11	
22	322.85	
23	599.07	

Figure 5. The user interface of MDP. The original imaging data is displayed at the center and the positions and sizes of the objects detected by SExtractor are overlaid as numbered circles. Long narrow rectangles indicate slits layout by user.

5. SUMMARY

The control software of FOCAS, FALCON, has been almost completed. It is designed as a network distributed, multi process system to achieve an asynchronous operation of the instrument. Since each process can be run independently and connects to each other with rather simple command interface using RPC, it is very easy to debug a process without stopping the whole system. This structure has the advantage of being relatively robust to bugs. The system will be tested in commissioning run of FOCAS.

Other than FALCON, we developed two off-line software systems, a multi-object slit plate design software, MDP, and a data reduction package for FOCAS performance verification.

ACKNOWLEDGMENTS

We thank T. Ezaki, S. Tsuda, T. Nishioka, K. Obata, A. Hirota, E. Miyano, K. Mukai, K. Ishibashi and A. Shiki of NIKON Co., Y. Komine and A. Tokuda of KNT Co., H. Nita and M. Sugita of Japan Laser Co. for developing the instrument. We also thank following NAOJ collaborators, T. Usuda, K. Omata, T. Nishimura, T. Noguchi, M. Yutani and R. Ogasawara. M. Y. are grateful to H. Maehara for his continuos encouragement. This project was supported in part by the Ministry of Education, Science and Culture of Japan through Grant-in-Aid 11691140.

REFERENCES

1. N. Kashikawa, M. Inata, M. Iye, K. Kawabata, K. Okita, G. Kosugi, Y. Ohyama, T. Sasaki, K. Sekiguchi, T. Takata, Y. Shimizu, M. Yoshida, K. Aoki, Y. Saito, R. Asai, H. Taguchi, N. Ebizuka, T. Ozawa, and

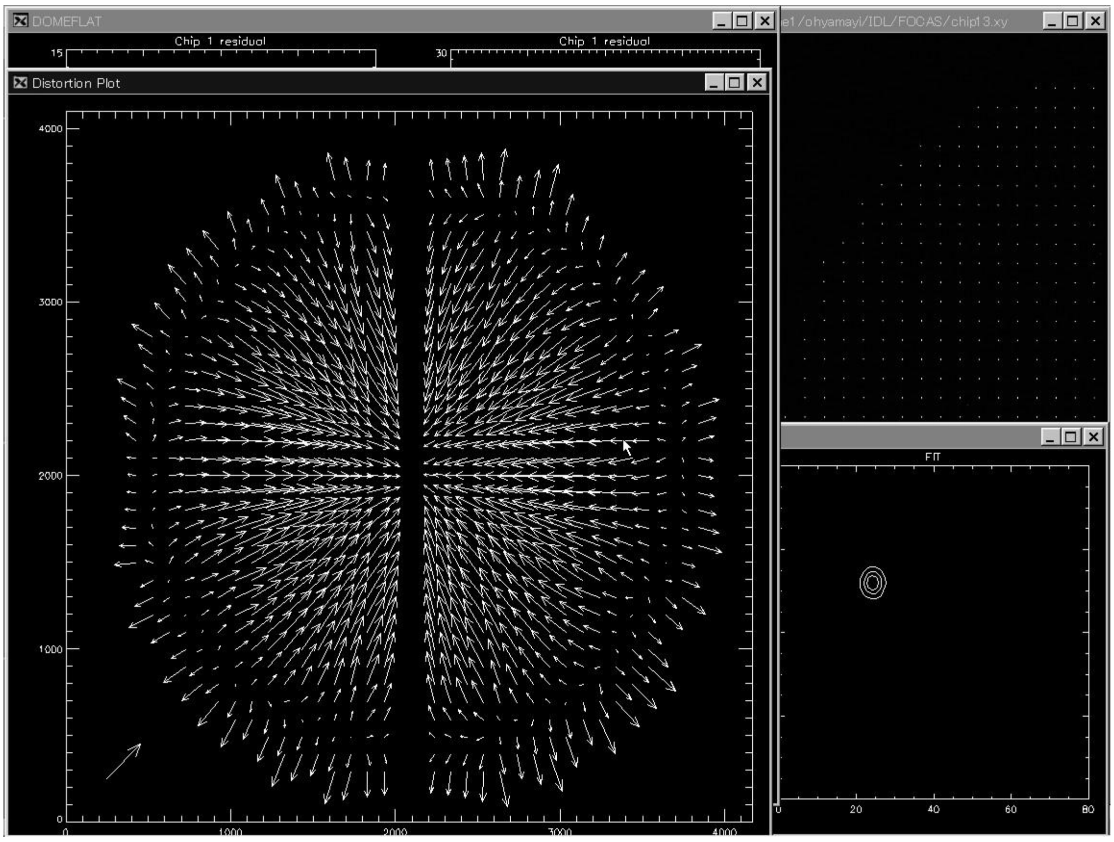


Figure 6. A sample display of the data reduction package for FOCAS performance verification. "Image Distortion" task is executing.

Y. Yadoumaru, "Test results of the subaru faint object camera and spectrograph focas," in *Astronomical Telescopes and Instrumentation 2000*, *Proc. SPIE* **4008-11**, 2000.

2. N. Kaifu, "Subaru telescope," in *Advanced Technology Optical/IR Telescopes VI*, L. M. Stepp, ed., *Proc. SPIE* **3352**, pp. 14–22, 1998.
3. T. Sasaki, M. Yoshida, Y. Shimizu, G. Kosugi, N. Kashikawa, Y. Yadoumaru, T. Takata, and M. Iye, "Constrol system of the focas instruments for the subaru telescope," in *Telescope Control Systems II*, H. Lewis, ed., *Proc. SPIE* **3112**, pp. 267–274, 1997.
4. G. Kosugi, T. Sasaki, T. Aoki, J. A. Kawai, N. Koura, and T. Kusumoto, "Subaru observation control system," in *Telescope Control Systems II*, H. Lewis, ed., *Proc. SPIE* **3112**, pp. 284–291, 1997.
5. T. Sasaki, G. Kosugi, J. Noumaru, T. Takata, Y. Mizumoto, R. Ogasawara, Y. Chikada, W. Tanaka, and J. A. Kawai, "Observation control system for the subaru telescope and its user interface," in *Observatory Operations to Optimize Scientific Return*, P. J. Quinn, ed., *Proc. SPIE* **3349**, pp. 427–434, 1998.
6. M. Sekiguchi, H. Nakaya, H. Kataza, and S. Miyazaki, "High-speed data acquisition system messia for subaru," *Experimental Astronomy* **8**, pp. 51–58, 1998.
7. T. Takata, R. Ogasawara, K. Kawarai, and T. Yamamoto, "Data archive and database system of the subaru telescope," in *Observatory Operations to Optimize Scientific Return*, P. J. Quinn, ed., *Proc. SPIE* **3349**, pp. 247–254, 1998.
8. G. Kosugi, T. Sasaki, Y. Mizumoto, T. Takata, J. A. Kawai, and Y. Ishihara, "Observation data set of the subaru observation software system," in *Observatory Operations to Optimize Scientific Return*, P. J. Quinn, ed., *Proc. SPIE* **3349**, pp. 421–426, 1998.
9. E. Bertin and S. Arnouts, "Sextractor: Software for source extraction.," *Astron. and Astrophys. Supple.* **117**, pp. 393–404, 1996.