Developing Infrared Array Controller with Software Real Time Operating System

Shigeyuki Sako[a], Takashi Miyata [a], Tomohiko Nakamura [a], Kentaro Motohara [a],
Yuka Katsuno Uchimoto [a], Takashi Onaka [b], Hirokazu Kataza [c]
[a] Institute of Astronomy, the University of Tokyo, 2-21-1 Osawa, Mitaka, Tokyo, Japan 181-0015;
[b]Department of Astronomy, the University of Tokyo, Bunkyo-ku, Tokyo, Japan 113-0033;
[c]Department of Infrared Astrophysics, Institute of Space and Astronautical Science, Japan Aerospace, Exploration Agency, Yoshinodai 3-1-1, Sagamihara, Kanagawa, Japan 229-8510

ABSTRACT

Real-time capabilities are required for a controller of a large format array to reduce a dead-time attributed by readout and data transfer. The real-time processing has been achieved by dedicated processors including DSP, CPLD, and FPGA devices. However, the dedicated processors have problems with memory resources, inflexibility, and high cost. Meanwhile, a recent PC has sufficient resources of CPUs and memories to control the infrared array and to process a large amount of frame data in real-time. In this study, we have developed an infrared array controller with a software real-time operating system (RTOS) instead of the dedicated processors. A Linux PC equipped with a RTAI extension and a dual-core CPU is used as a main computer, and one of the CPU cores is allocated to the real-time processing. A digital I/O board with DMA functions is used for an I/O interface. The signal-processing cores are integrated in the OS kernel as a real-time driver module, which is composed of two virtual devices of the clock processor and the frame processor tasks. The array controller with the RTOS realizes complicated operations easily, flexibly, and at a low cost.

Keywords: array controller, detector, infrared, real-time operating system

1. INTRODUCTION

Sensitivity in astronomical observations is determined by collecting abilities of stellar photons and its utilization efficiency. The latter depends largely on the number of detector pixels, which handle the collected photons concurrently. A large format array is a powerful tool to obtain wide field images. Several survey programs are ongoing and planned with taking advantage of the large format array, e.g., the Suprime-Cam visible camera[1] on the Subaru telescope and the WFCAM near infrared camera[2] on the United Kingdom Infrared Telescope. The large format array improves the observing efficiency in spectroscopy as well as in imaging. It covers a wider spectral range in a single exposure, and is especially significant for high dispersion spectrographs, e.g., the IRCS near infrared camera and spectrograph[3] on the Subaru telescope. In multi-object spectrographs, the large format array can obtain spectra of many stellar objects in a field of view at a time, e.g., the MOIRCS multi-object near infrared camera and spectrograph[4] on the Subaru telescope. Meanwhile, a larger format array produces larger overhead time attributed to readout and data processing. When the overhead time is not negligible relative to an exposure time, it becomes prominent as a dead time of the observations. This is crucial for short exposure time in broad-band infrared observations from the ground, typical exposure time is few dozen of milliseconds and several seconds in the mid- and near-infrared, respectively. Thus, the advantages of the large format array are fully utilized only when the dead time is reduced to a minimum.

An array controller is a key device to determine the observation efficiency. The exposure is suspended while reading and transferring the data to storages in the exposure-then-readout operation. To reduce the dead time, the data transfer should be completed as fast as possible and then the next exposure be quickly resumed. In the exposure-during-readout operation, the array produces the frame data constantly during the exposure, and so the data stream should be transferred at a rate faster than the frame rate without a break. Organization of the data acquisition processes is important to improve the observation efficiency, too. The dead time is reduced by compression of the frame data and parallel processing between the exposure, the data transfer, and primary analysis. In order to perform well organized multiple processes, it is required to monitor their statuses and harmonize conflicts between the processes in real time.

An array controller with multitasking and real time capabilities has been developed with dedicated processors including digital signal processors (DSPs), field programmable gate arrays (FPGAs), and complex programmable logic devices (CPLDs), e.g., the COMICS array controller[5] and the Messia V array controller[6]. The processors perform complicated tasks in parallel at a high speed according to logics programmed in hardware description languages. They are widely used as a main or auxiliary processor in a large variety of control system, but there are some problems in developing a high-performance array controller. The FPGA and the CPLD devices have not enough resources to execute large size programs. The DSP device has problems with inflexibility and high cost, while it is accessible to large on-board and/or external memories.

In this study, we have developed a high-speed and flexible array controller for the large format infrared array with a software real-time operating system (RTOS) instead of the dedicated processors. In Section 2, architecture and composition of our system will be described. Experimental tests and examples of implementation will be mentioned in Section 3.

2. DESIGN FOR ARRAY CONTROLLER

2.1 Architecture

We have designed the array controller with an architecture based on resources in PCs mounted on an observation instrument. A recent PC is equipped with one or several multi-core CPUs and large capacity of memories. The resources are sufficient to control the arrays and to process a large amount of frame data even while performing an operating system and some application programs. By using the RTOS on the high-performance PC, the real-time processing can be performed without dedicated processors such as DSPs, FPGAs, and CPLDs. The RTOS executes hardware processes as higher-priority tasks in preference to other software processes. A time jitter and latency in an execution of the hardware process are much lower than those on a non-real time operating system. However, the time jitter is not small enough to control the array directly because a scheduler of the RTOS produces a time jitter of a few microseconds. A digital I/O board with direct memory access (DMA) controllers helps to attain a time jitter of typically 10 to 100 nanoseconds, which is smaller than an acceptable value to control the array. The DMA controller transfers the control (clock) data and the frame data from and to the main memories on the PC without loads to the CPU. With the combination of the RTOS and the digital I/O board, a high-speed and real-time array controller can be established without the dedicated processors.

This architecture has no substantial limitations of the number of logics unlike the FPGA and the CPLD devices because the signal-processing cores are integrated to the operating system as driver modules, which have an access to huge memory resources in a kernel space. The memory space can be utilized from user applications directly and quickly, too. Since the driver module is programmed as a general application, we can make up complicated signal-processing cores easily and flexibly. The fact that the composition is based on general-purpose devices like a common used PC and a digital I/O board is also one of the advantages. It makes improving and replacing the system easier and leads to low maintenance cost. A wide variety of digital I/O boards are manufactured by several vendors at a relatively low price. Recently, several RTOSs are distributed for free or at reasonable prices, too, e.g., RTAI[7] and RTLinuxFree[8].

2.2 Composition

We have developed the array controller, the TAO* array controller (TAC) system, for the large format infrared arrays using the RTOS. The composition diagram of the TAC system is shown in Figure 1. A PC equipped with a dual-core CPU; Intel Pentium D processor, and main memories of two Gbyte in total is used as a main computer, and a Linux kernel with RTAI (the Real-Time Application Interface for Linux)[7] extension is adopted as the RTOS. RTAI is an extension of Linux and supports hardware real-time operation through interrupt control between the hardware and the operating system. Interrupts and tasks required for the real-time processing are performed by the real-time core, while others are forwarded to the non-real time operating system. The Linux operating system runs as the lowest priority task. Interface Co. Ltd, PCI-2772c is adopted as an interface board[9]. This is a general-purpose parallel I/O board with 32 bit, 33 MHz PCI bus supporting the DMA transfer through fast-in-fast-out (FIFO) buffers. Parallel data is transferred from and to front-end circuits with 32 data lines in synchronization with internal and external base clocks, up to 20MHz. Four interrupt lines are equipped, too. The data and the interrupt lines are evenly shared by input and output. The I/O board is

connected to an A/D converter board with a parallel cable or an RJ-45 cable in the bus low voltage differential signaling (BLVDS) protocol.

The digital processing is managed by a real-time driver module, called a TAC module, integrated in the kernel with the RTAI extension. One core of the dual-core CPU is allocated for the TAC module. By doing this, the loads of the Linux operating system attributed to the real-time processing is reduced drastically because the single CPU core works as a dedicated processor for the array control like DSPs, FPGAs, and CPLDs. The TAC module performs sorting, co-adding, and a least square fitting of the frame data while handling hardware interrupts from the digital I/O board. The processed data is transferred to the hard disk drive by user applications. In addition, the TAC module generates the clock data for the array control by decoding source data, which is loaded to the main memory beforehand.

*the University of Tokyo Atacama Observatory (PI: Yuzuru Yoshii)[10]

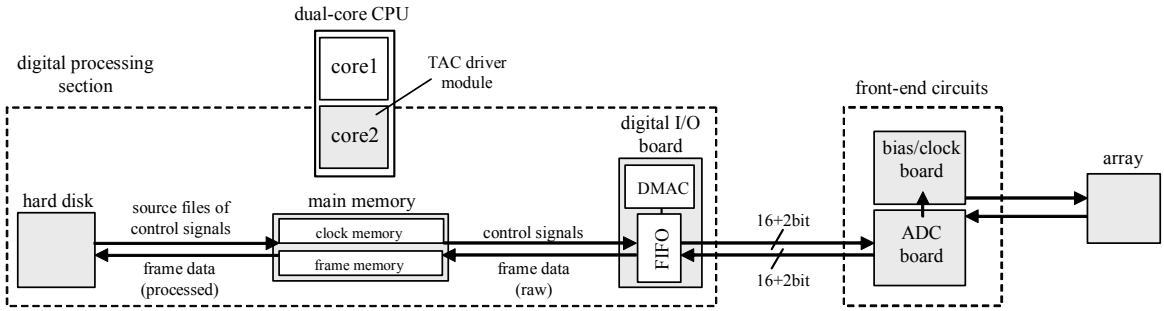


Figure 1. Composition diagram of the TAC system. The digital processing section is managed by the TAC driver module to which one core of the dual-core CPU is allocated. The frame data is transferred from the front-end circuits to the main memory via the digital I/O board. The stored frame data is processed by the TAC module on the memory, and then is transferred to the hard disk drive. The TAC module also generates the clock data for the array control by decoding source data.

2.3 TAC real-time driver module

The TAC real-time driver module, *tac_core.ko*, is composed of the clock processor (CLP) task, the frame processor (FRP #1 and 2) tasks, and several handlers for the interrupts and kernel system-calls as shown in Figure 2. The interrupt handler is performed as the first priority task in the real time module. The second to fourth priorities are allocated to the CLP and the FRP #1 and 2 tasks, respectively. The TAC module operates the digital I/O board with the driver module *i2772c.ko*, which is exclusively developed for the PCI-2772c board. Memory spaces are allocated for the CLP and the FRP tasks in the main memory. User processes access to the TAC module with system calls of ioctrl and mmap, and deals with statuses of the module via the real-time FIFO (RTFIFO) interfaces of RTAI. The CLP task generates control (clock) data for the array control. The FRP tasks handle and process the frame data in real time. Details about the CLP and the FRP tasks are described in Section 2.3.2 and 2.3.3, respectively. When receiving a transfer request from the TAC module via the RTFIFO, the user application begins copying the processed frame data in the FRP memory to the user space, and then the data is saved to the hard disk drive as a FITS file.

2.3.2 Clock processor (CLP) task

The clock data is transferred from the CLP memory to the front-end circuits by the DMA controller in synchronization with the internal clock on the digital I/O board. The DMA controller handles the clock data in small units, and each source address and size of the data unit are registered in a scatter-gather (SG) FIFO buffer. When a frequency of the internal base clock is set to 5 MHz, which is a typical rate used for control of the infrared array, the SG-FIFO data and the clock data are depleted in short time of less than a few dozen milliseconds. To transfer the clock data to the front-end circuits continuously, it is required to replenish new data in the SG-FIFO buffer and the CLP memory before the depletion. In the TAC system, the CLP task performs the replenishment processes whenever it receives the interrupt

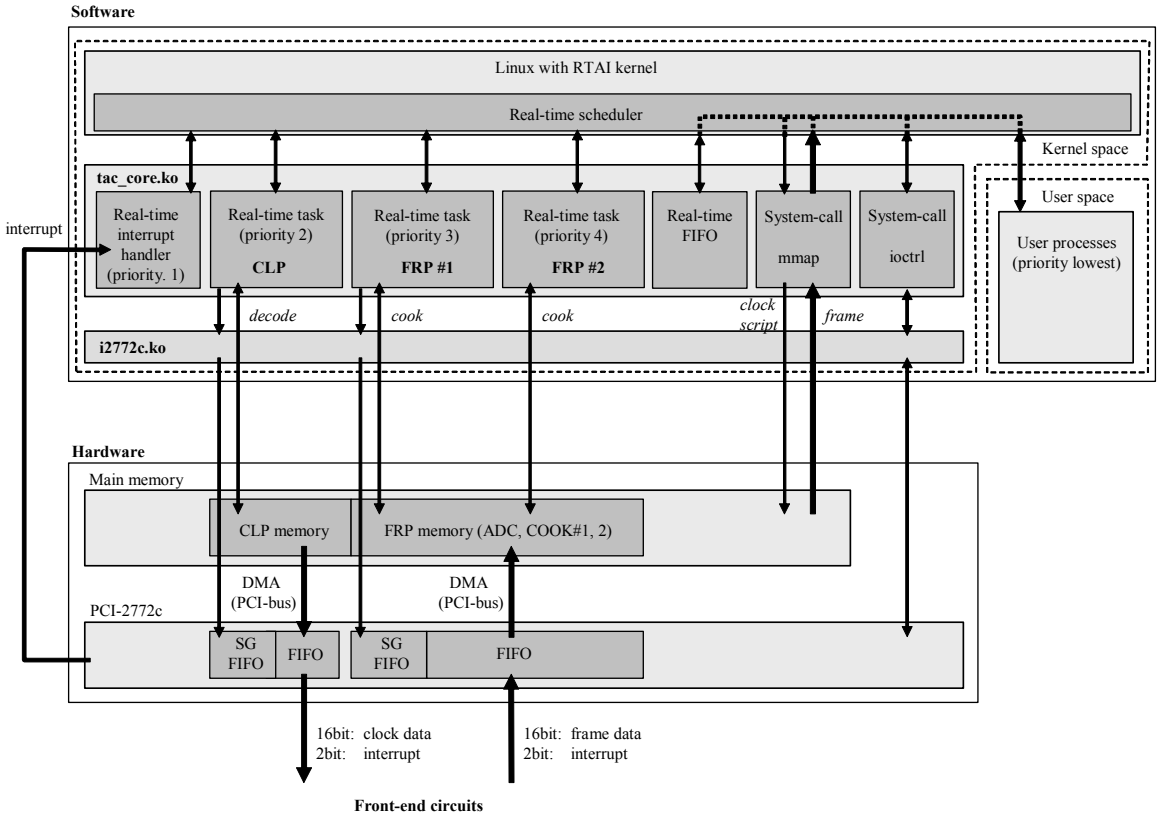


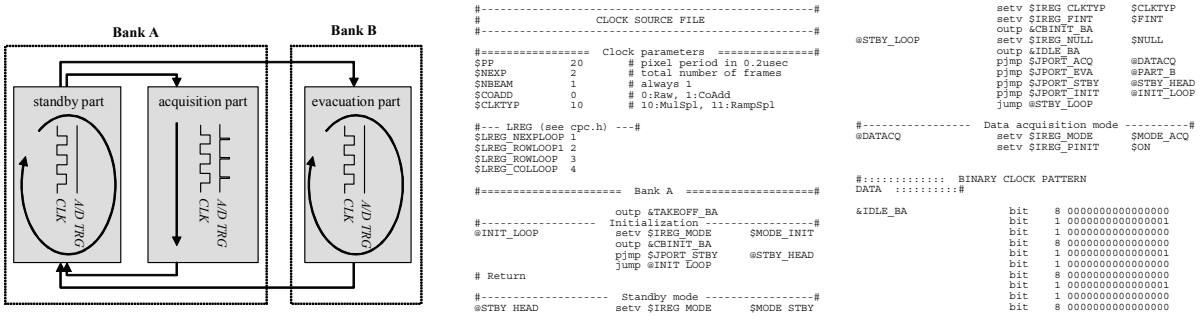
Figure 2. Diagram of data flow and relationship between tasks and devices in the TAC system. The software part and the hardware part are shown separately. The TAC module and the device driver module *i2772c.ko* are integrated in the Linux kernel with RTAI. The TAC module's tasks are scheduled by the real-time core of RTAI. The digital I/O board is operated by the TAC module with *i2772c.ko*. The TAC module is composed of the clock processor (CLP) task, the frame processor (FRP #1 and 2) tasks, and several handlers for the interrupts and kernel system-calls. The CLP task generates clock data for the array control. The FRP tasks handle and process the frame data in real time. The user processes access the TAC module with the system calls of ioctrl and mmap.

signal of the transfer requirement from the digital I/O board. Because the response time of the real-time interrupt handler is much less than the time scale of the depletion as described in Section 3.2, the CLP task can certainly replenish the clock data in the SG-FIFO buffer and the CLP memory before they are depleted.

In many cases, array controllers generate all of clock data necessary for operations in a clock memory before starting exposures. This is a simple way, but huge memory space is required especially when operating it with the exposure-during-readout mode or the multiple-sampling mode. The exposure time is limited by overflow of the clock memory. Meanwhile, the CLP task generates clock data necessary for operation in near feature by decoding the source data in real-time. This requires a little memory space for the clock data even in continuous operation. The source data is described in original assembler-like codes, and is loaded in a part of the CLP memory. The CLP task supports definitions of constants and variable parameters, and structures of loop, conditional branch, sub-routine, and address jump. The user application can dynamically control the conditional branch by accessing the variable parameters with the ioctrl system-call.

The CLP task has a similar architecture to a general operating system which is composed of infinite loops, conditional branches, and registers. Therefore, various clock patterns can be designed with the CLP task as programming general application softwares. We have developed the source data for operating the Si:Sb BIB 128 × 128 mid infrared array. Performances of this array are quit sensitive to the array temperature, which is cooled down at 6 to 8 K. To keep the

array temperature constant, we repeat reading the array during observations without changing the basic pattern of the clock. This operation method prevents variation of power dissipation of the array[5]. The source data for the Si:Sb BIB 128 × 128 array is composed of three parts; standby part, acquisition part, and evacuation part as shown in Figure 3. In the exposure, the CLP task generates the clock data by decoding the source data in the acquisition part. When the exposure is finished, the decoding target is switched to that in the infinite loop of the standby part automatically. The source data in the standby part is similar to that in the acquisition part except a trigger signal for the A/D converter. Therefore, there is no difference in the power dissipation of the array between in the acquisition part and in the standby part. When the decoding target is switched from the standby part to the acquisition part, the exposure is started again. The source data in the evacuation part is the same as that in the standby part. This structure allows us to exchange the data source without stopping the output of the clock signal. First, the new source data is loaded to both the standby and the acquisition parts, Bank A, after the decoding target is switched to the evacuation part. Next, the new data is loaded in the evacuation part, Bank B, after the decoding target is returned to the standby part. With this way, the detector temperature can be kept constant.




```
#-----------------------------------------------------#
#                 CLOCK SOURCE FILE                   #
#-----------------------------------------------------#

#=================  Clock parameters  ===============#
$PP             20      # pixel period in 0.2usec
$NEXP           2       # total number of frames
$NBEAM          1       # always 1
$COADD          0       # 0:Raw, 1:CoAdd
$CLKTYP         10      # 10:MulSpl, 11:RampSpl

#--- LREG (see cpc.h) ---#
$LREG_NEXPLOOP 1
$LREG_ROWLOOP1 2
$LREG_ROWLOOP  3
$LREG_COLLOOP  4

#=====================  Bank A  ====================#

                    outp &TAKEOFF_BA
#-----------------  Initialization -----------------#
@INIT_LOOP          setv $IREG_MODE      $MODE_INIT
                    outp &CBINIT_BA
                    pjmp $JPORT_STBY     @STBY_HEAD
                    jump @INIT_LOOP
# Return

#--------------------  Standby mode -----------------#
@STBY_HEAD          setv $IREG_MODE      $MODE_STBY
                    setv $IREG_CLKTYP    $CLKTYP
                    setv $IREG_FINT      $FINT
                    outp &CBINIT_BA
@STBY_LOOP          setv $IREG_NULL      $NULL
                    outp &IDLE_BA
                    pjmp $JPORT_ACQ      @DATACQ
                    pjmp $JPORT_EVA      @PART_B
                    pjmp $JPORT_STBY     @STBY_HEAD
                    pjmp $JPORT_INIT     @INIT_LOOP
                    jump @STBY_LOOP

#-----------------  Data acquisition mode ----------#
@DATACQ             setv $IREG_MODE      $MODE_ACQ
                    setv $IREG_PINIT     $ON

#:::::::::::::  BINARY CLOCK PATTERN
DATA  ::::::::::#

&IDLE_BA                bit    8 0000000000000000
                        bit    1 0000000000000001
                        bit    1 0000000000000000
                        bit    8 0000000000000000
                        bit    1 0000000000000001
                        bit    1 0000000000000000
                        bit    8 0000000000000000
                        bit    1 0000000000000001
                        bit    1 0000000000000000
                        bit    8 0000000000000000
```

Figure 3. (*left*) Loop structures of the clock source data designed for the Si:Sb BIB 128 × 128 mid infrared array. The source data is composed of three parts. The pattern of the clock data is similar to each other except the A/D trigger signal. Details are described in the text. (*right*) Code of the clock source data. It is described in an original assembler-like language. Only a part of the code is shown.

2.3.3 Frame processor (FRP) task

The FRP memory, which is allocated to the FRP tasks, is composed of two parts; the ADC and the COOK parts as shown in Figure 4. The raw frame data is transferred from the front-end circuits to the ADC part by the DMA controller in synchronization with an external base clock and a data strobe clock. The DMA controller handles the frame data in small units, and each destination address and size are registered in a SG-FIFO buffer. As in the case of the CLP task, the FRP #1 task replenishes the SG-FIFO buffer whenever it receives the interrupt signal of the replenishment requirement from the digital I/O board. To remove a limitation on the data size, ring-buffer structures are formed in the ADC and the COOK parts.

When an external frame clock, which means that it is completed to transfer a single frame to the ADC part, is received from the front-end circuit, the FRP #1 task begins copying the frame data from the ADC part to the COOK part. The copy process varies with the operation mode of the FRP #1 task. In a raw mode, the frame data in the ADC part is copied straight to the COOK part. This process produces a carbon copy of the ADC part in the COOK part. In a co-add mode, the FRP #1 task copies it to the COOK part with co-adding specified number of the frames. Because the co-adding process compresses the data flow, it leads to drastic reduction of the load of the data bus as well as cutting-down the data transfer time. In a multiple-sampling mode, a specified number of frames are taken in the beginning and in the end of the exposure, and the frames in each set are co-added with the negative and the positive signs, respectively. A multiple-sampling method leads to reduction of the readout noise of the frame data. The FRP task supports the ramp-sampling method, too, in which a specified number of frames are read nondestructively with a certain interval time during the exposure. The frame set is fitted with the linear least square algorithm to derive the signal value from the increasing rate of the pixel count. The ramp-sampling method allows us to not only obtain images with lower readout noise than those

with the multiple-readout method, but also prevent saturation of the signal attributed to bright sources. Assuming incident flux and the sampling interval are constant in time during the exposure, the signal value for a certain pixel in the ramp-sampling method is expressed as

$$signal = \frac{N-1}{n-1}\,\frac{12}{n(n+1)}\,\sum_{i=1}^{n}\left(i-\frac{n+1}{2}\right)y_i\,, \tag{1}$$

where y_i is the pixel count of the i th sampling, N the total number of the sampling, and n the number of the valid data without saturation. However, this fitting calculation cannot be performed on the memory in real-time because the parameter of n cannot be predicted in advance. Additionally, it needs much CPU and memory resources because of including several floating-point operations. In the TAC system, the FRP task produces three kinds of images with simple integer calculations described in Equation (2) to (4) in real-time, and then the user application derives the signal value by combining them with Equation of (5) after creating the FITS files,

$$S_1 = \sum_{i=1}^{n} y_i \tag{2}$$

$$S_2 = \sum_{i=1}^{n} i y_i \tag{3}$$

$$S_3 = \mathrm{n} \tag{4}$$

$$signal = \frac{12(N-1)}{S_3(S_3^2-1)}\left(S_2-\frac{S_3+1}{2}S_1\right). \tag{5}$$

Three work spaces are allocated to the S_1, S_2, and S_3 calculations in the COOK part in the ramp-sampling mode. The S_3 parameter is counted up when the pixel count is in linearity response range.

The frame data in the COOK part is copied to the hard disk drive by the user application before the ring-buffer is filled. When the exposure is completed or untransferred data occupies a certain amount of the COOK part, the FRP #2 task informs the user space of a transfer requirement via the RTFIFO buffer. When the requirement in the RTFIFO is detected, the user application begins copying the COOK data to the user space with the mmap system call, and then the data is saved to the hard disk drive as a FITS file. Newly transferred data is appended to the saved FITS file whenever the transfer request signal triggered by on-going exposure is received.

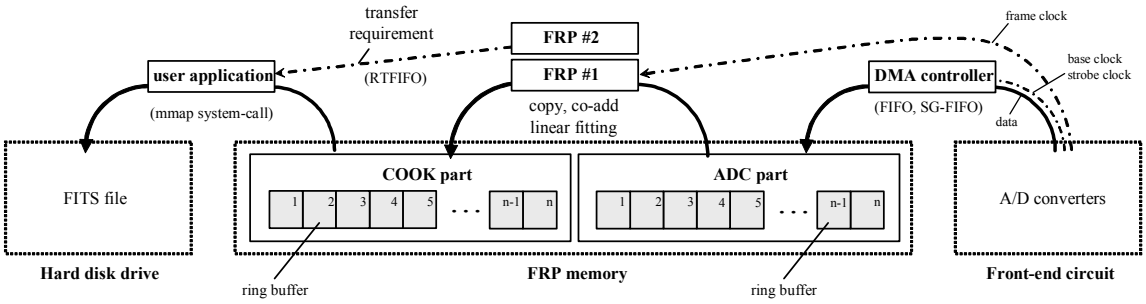


Figure 4. Diagram of data handling. The frame data is transferred from the front-end circuit to the ADC part in the FRP memory by the DMA controller on the digital I/O board. The FRP #1 task performs processing of the frame data with copying it from the ADC part to the COOK part. When an exposure is completed or untransferred data occupies a certain amount of the COOK part, the FRP #2 task informs the user space of a transfer requirement via the RTFIFO. The user application produces a FITS file from the frame data in the COOK part, and saves it in the hard disk drive.

3. APPLICATION AND PERFORMANCE

3.1 Application

The TAC system has capabilities to control various kinds of the infrared arrays and the CCDs. In particular, it is effective in control of the infrared arrays requiring the real-time process in parallel with the exposure. We have applied the TAC system to operations of the Si:Sb and the Si:As BIB 128 × 128 mid infrared arrays for the MAX38 camera[11] and of the HAWAII-2 2k × 2k near infrared array for the ANIR camera[12]. The TAC system will be integrated in the MIRSIS mid infrared spectrograph[13] to control the Si:As BIB 320 x 240 array.

The mid-infrared arrays are required to be operated with the exposure-during-readout mode in broad-band observations from the ground. The frame data is composed of a large number of frames with a short exposure time less than 50 milliseconds. It is obtained with switching the beams every several 100 milliseconds, which is so-called chopping observations. In the MAX38 system, the FRP tasks perform the compression processing of the complicated frame set in real-time, and produce the FITS files without a break. This drastically reduces the dead time attributed to the readout. The base clock of 5 MHz is applied to the input of the frame data, and a memory space of 104 Mbytes, which corresponds to 1,300 frames of the 128 × 128 array, is allocated to the FRP tasks. As described in Section 2.3.2, the CLP task generates the clock data by decoding the source data having the infinite loop structures, which are necessary for keeping the array temperature constant. The base clock of 5 MHz is applied to the output of the clock data, and a memory space of 24 Mbytes is allocated for the CLP task.

The ANIR system adopts the ramp-sampling method for readout of the HAWAII-2 near infrared array to obtain narrow band images with low readout noise. The FRP tasks perform the linear least square fitting to the ramp-sampling data on the FRP memory in real-time. The base clock of 5 MHz is applied to the input of the frame data, and a memory space of 104 Mbytes, which corresponds to 52 frames of the 1k × 1k array, is allocated to the FRP tasks. We designed the clock source data for the HAWAII-2 array with the infinite loop structures to reset the array pixels during the idle state. The reset idling clock suppresses persistent images after bright sources are observed. The base clock of 5 MHz is applied to the output of the clock data, and a memory space of 24 Mbytes is allocated for the CLP task.

3.2 Performance

The real-time performances of the RTOS are crucial to design the array controller. We measured latencies of the response to the hardware interrupt signal in the RTAI system. The minimum and the maximum latencies are listed in Table 1. They mean that the jitter time is typically a few micro seconds. To complete the interrupt response certainly in the array control system, the occurrence interval of the interrupts should be set long enough both for the jitter time and for the execution time of the interrupt response itself. In the TAC system, an amount of the tasks of the modules and the frequencies of the base clocks on the digital I/O board are adjusted to satisfy the requirements.

The maximum throughput of the TAC system depends on performances of the data buses between the front-end circuits and the hard disk drive. The digital I/O board PCI-2772c has the capability to transfer data between the FIFO buffers on the board and the front-end circuits at the maximum rate of 320 Mbits/second. The FIFO data is transferred from and to the main memory, DDR2 667 PC5300, through the 32bit 33MHz PCI bus. The peak transfer rate of the memory bus is 42 Gbits/second. While the peak transfer rate of the PCI bus is 1 Gbits/second, the actual throughput of the PCI bus slows to less than half the peak rate in ordinary use. Therefore, a bottleneck of the data transfer is at the PCI bus unless the FRP tasks perform heavy processing. Here, the throughput of the hard disk drive, whose peak transfer rate is typically less than 100 Mbits/second, is taken no account because the frame data can be compressed in the main memory by the FRP tasks. We operate the TAC system at a throughput of 80 Mbits/second in both the MAX38 and the ANIR systems. To improve the throughput, it is necessary to extend the data bus of the digital I/O board, which is the bottleneck of the data transfer. When a digital I/O board with the PCI Express ×16 bus, whose peak transfer rate is 40 Gbits/second, is applied, the actual throughput of the TAC system will be increased up to more than 1 Gbits/second. When a much larger throughput is required, the array control system should be established with multiple PCs. Because

the TAC architecture is based on the software RTOS without the dedicated processors, it can realizes such complicated operations with the multiple PCs easily and flexibly.

Table 1. Latencies of the response to the hardware interrupt signal. The minimum and the maximum values of about one minute measurement are listed. The values were measured with a test program distributed together with the RTAI extension.

CPU name	CPU clock	cores	OS	min. latency	max. latency
Pentium 4 (530)	3.0GHz	1	Linux2.6.17+RTAI3.4	0.0μsec	3.6μsec
Pentium D (830)	3.0GHz	2	Linux2.6.17+RTAI3.4	0.0μsec	3.2μsec
Pentium Core2Duo (E6700)	2.66GHz	2	Linux2.6.20+RTAI3.5	0.0μsec	0.5μsec

4. SUMMARY

We have developed the array controller TAC for the large format infrared arrays with the RTOS without any use of the dedicated processors. In this system, a Linux PC equipped with RTAI extension and a dual-core CPU is used as a main computer, and one of the CPU cores is allocated to the real-time processing. The digital I/O board with DMA functions is used for the I/O interface. The signal-processing cores are integrated in the OS kernel as a real-time driver module, which is composed of several handlers for the interrupts and two virtual devices of the clock processor (CLP) and the frame processor (FRP) tasks. The CLP task generates the clock data for the array control, and the FRP tasks handle and process the frame data in real time. We operate the TAC system with a transfer data rate of 80 Mbits/second in the systems of the MAX38 mid infrared camera and the ANIR near infrared camera. The architecture of the TAC system realizes the complicated operations in easily, flexibly, and low cost.

This research was supported by a Grant-in-Aid for Scientific Research on Priority Areas from the Ministry of Education, Culture, Sports, Science and Technology of Japan. We are grateful to all of the staff members of the TAO project. S. S. is financially supported by the Japan Society for the Promotion of Science (18-9936).

REFERENCES

[1] Miyazaki, S., et al., “Subaru Prime Focus Camera --- Suprime-Cam,” PASJ, 54, 833-853 (2002)
[2] Henry, David M. et al., "Wide-field camera for the United Kingdom Infrared Telescope," Proc. SPIE 4008, 1325-1333 (2000).
[3] Kobayashi, N., et al., “IRCS: Infrared Camera and Spectrograph for the Subaru Telescope,” Proc. SPIE 4008, 1056-1066 (2000)
[4] Ichikawa et al., "MOIRCS: Multi-Object Infrared Camera and Spectrograph for SUBARU," Proc. SPIE 6269, 38-49 (2006)
[5] Sako, S. et al., "Developing the COMICS array controller," Proc. SPIE 4445, 119-129 (2001).
[6] Nakaya, H. et al., "New Focal Plane Array Controller for the Instruments of the Subaru Telescope," PASP, 118, 841, 478-488 (2006).
[7] WWW home of the RTAI; https://www.rtai.org/
[8] WWW home of the RTLinuxFree; http://www.rtlinuxfree.com/
[9] WWW home of Interface co. ltd.; https://www5.interface-world.com/
[10] Yoshii, Y. et al., "Tokyo Atacama Observatory Project", Proc. IAU 8th Asian-Pacific Regional Meeting II, 35-36 (2002)
[11] Miyata, T. et al., "A new midinfared camera for ground-based 30 micron observations: MAX38," Proc. SPIE, in this conference.
[12] Motohara, K. et al., "ANIR: Atacama near infrared camera for Paschen alpha imaging," Proc. SPIE, in this conference.
[13] Okamoto, Y. K. et al., " Development of a test N-band image slicer: optical design," Proc. SPIE 6269, 172-182 (2006)