SOFTWARE ARCHITECTURE AND WAVEFORM APPLICATIONS OF SOFTWARE DEFINED RADIO BASED ON SCA V2.2

Yusuke KANAHASHI; Isao TESHIMA;
Satoru NAKAMURA; Mitsuyuki GOAMI; Takuzo FUJII
Communication Systems Developing Group, Hitachi Kokusai Electric Inc.,
Tokyo, Japan; email: kanahashi.yusuke@h-kokusai.com

ABSTRACT

In "Development of Software Radio Prototype" presented during the 2002 SDR Forum Technical Conference, we reported on a prototype software defined radio (SDR), providing various types of analogue/digital modulation, two full duplex channels, and a frequency range of 2-500MHz. Last year's effort evaluated hardware design approaches for multi-band, multi-mode SDR [1]. In the current article, we report on a new prototype with upgraded hardware and adoption of the Joint Tactical Radio System (JTRS) Software Communication Architecture (SCA) v2.2 [2], examining our software architecture approach and *Waveform Applications* (the word in italics means a term in SCA v2.2 hereafter).

For flexibility and expandability, our software architecture includes: *Operating Environment (OE), Core Framework (CF) Service*, and *Waveform Applications* that inherit the *CF*. For the *OE*, POSIX compliant embedded real time Linux is installed in each CPU, and the "software bus" employs CORBA. "*CF Service & applications"*, defined by SCA, are allocated to each OS of CPU. The *Waveform Application* is built by "*Domain Profiles and Application Factory*" in the *OE*, defined by SCA, using software downloaded to the Control module. Each element of the *Waveform Application* implements the software interface that inherits "*Base Application Interface*" by SCA.

Our research platform demonstrates the effectiveness of an SDR based on SCA v2.2 for flexibility and expandability of *waveform applications*.

1. INTRODUCTION

In order to realize high flexibility and adaptability in the SDR, it is necessary to develop both the hardware and software technologies that assure such performance. The previous prototype was developed to realize such a hardware platform. The new prototype was developed to adopt the software architecture of SCA v2.2, examining flexibility of *Waveform Applications* for the software execution environment and the expandability for functions. That is: building *OE*, the following were developed based on SCA v2.2: *Core Framework Services & Applications* necessary for *Waveform Applications* building, and *Waveform Applications* which inherit *Base Application Interface*. Software for data transmission and a peripheral equipment control utility were also developed, examining: flexibility and adaptability of *Waveform Applications*, and simultaneous operation of peripheral equipment control with the *Waveform Applications.* This paper reports on these topics.

2. BASIC DESIGN CONCEPT

In addition to the concepts inherited from the previous prototype: 2-500 MHz coverage, four full-duplex channels, wideband signal processing using higher IF frequency, a CPU in each main module, and appropriate use of FPGA and DSP: the following aspects were considered for basic design concept.

1) Building of SCA v2.2 software architecture on the CPU of each hardware module.
2) Operation flexibility and function expandability of application software.

3. SPECIFICATIONS OF PROTOTYPE

Based upon the above basic concepts, the specifications were fixed as shown in Table 1.

Table 1. Specifications of the prototype

RF range	2-500MHz
Waveform	SSB (USB, LSB), AM, FM, BPSK, QPSK, 8PSK, 16QAM
Number of channels	4-channels of full duplex
Radio relay	Repeat/Bridge
Frequency accuracy	<0.1ppm
IF frequencies	Tx:25MHz Rx:70MHz
Dynamic range	14bits (Quantize bits)
Rx IF sampling freq.	40MHz (Under sampling)
Tx IF sampling freq.	100MHz (4x over sampling)
Signal processing	FPGA: Quadrature modem DSP: Baseband modem
System bus	Signal/control Unit: cPCI RF Unit: Original (cPCI Form factor)
Interface	Serial I/O, Analog I/O, Digital I/O, Ethernet (100 BASE-TX)
Software environment and applications	Based on SCA v2.2

4. SYSTEM DESIGN

As shown in Fig.1, this prototype SDR is connected to a Control terminal and a VoIP terminal through Ethernet using TCP/IP protocol. The Control terminal controls SDR operation and operates as a server using FTP protocol when an application is downloaded to the SDR. The VoIP terminal is used for VoIP telephone and for data transmission, which is also available through RS232C. The peripheral equipment such as Power amplifier is controlled by SDR through RS232C.

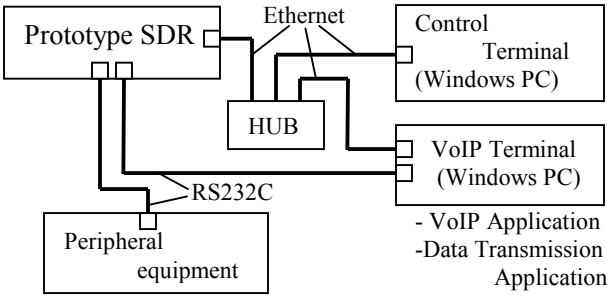

Figure 1. System Overview

5. HARDWARE

As shown in Fig.2, the SDR hardware consists of an RF unit and a Signal processing/control unit. Each unit contains several 6U (Euro card) compact PCI (cPCI) modules. The hardware structure of the previous prototype was upgraded for the following point:

1) Accommodation capacity was increased by absorbing the function of Data converter, which was a separated module in the previous prototype, into the Quadrature modem module.

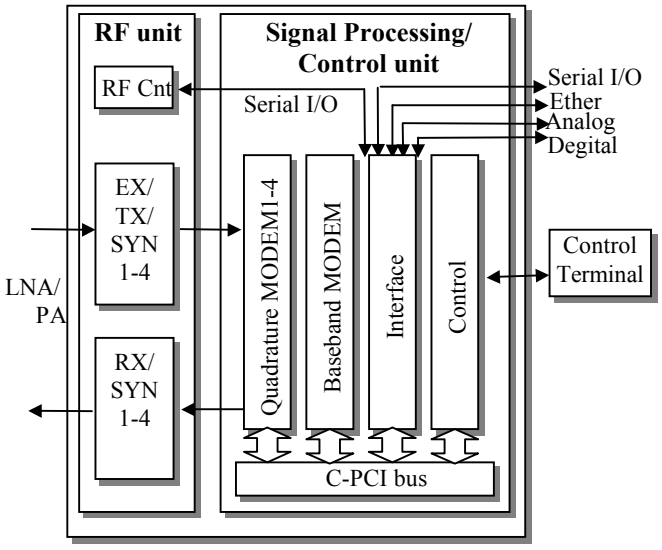

Figure 2. Block diagram of the prototype

2) Ethernet interface for data input/output was added to the Interface module.

3) Repetition speed of the radio frequency switching was accelerated due to the control made by RF control module added to the RF unit, which was made by Signal processing/ control unit in the previous prototype.

5.1 Signal processing/control unit

The Signal processing/control unit contains the following modules; Quadrature modem, Baseband modem, Interface, and Control. Up to four-channels-Quadrature modem can be installed. Each module is connected to each other by PCI bus, and provides a CPU.

The Quadrature modem processes quadrature modulation/ detection, sampling rate conversion, and filtering, using FPGAs. It also processes, D/A conversion for the transmitted IF signal and filtering, and filtering and A/D conversion of the received IF signal. The CPU performs data transmission/reception with the FPGA configuration using the PCI bus, and the FPGA configuration data can be downloaded through the CPU.

The Baseband modem processes multiple channel modulation/demodulation process using four floating points DSP devices. The processing can be flexibly assigned to each of the four DSPs. The CPU relays transmitting/receiving data between DSP and PCI bus, and the software for the DSP can be downloaded from the Control terminal through the CPU. Since an individual DSP is assigned for each channel, even if processing of either channel is under execution, a program can be downloaded to another channel.

The Interface module provides: RS232C I/O port for RF unit control, and Ethernet interface for the data input/output. In addition, it provides analog I/O and digital I/O port interfaces allowing various types of legacy radio to be replaced with this SDR.

The Control module that provides the main control function of this prototype occupies the cPCI system slot. It provides an Ethernet interface for reception of control signal from the Control terminal, and transmission of response signal and status information to the Control terminal.

5.2 RF Unit

The RF unit has four channels capacity. Each RF channel consists of modules: Transmitter/Exciter with Synthesizer, and Receiver with Synthesizer. An RF control module that includes a CPU is provided to control these modules in the RF unit. Receiving control commands from the Control terminal via Interface module in the Signal processing/control unit, the RF control module distributes the command data to corresponding modules in the RF unit. Also, receiving status information from modules in the RF unit, RF control module transmits it to the

Control terminal via the opposite route. RF control module is connected to the Interface module by RS232C, and to the other modules by the original bus.

In the Transmitter module, triple frequency conversion, with a third IF higher than the transmitting frequency, is employed for frequency conversion from first IF of 25MHz into RF signals of 2-500MHz. In the Receiver module, double super heterodyne design, with a first IF frequency above the receiving frequency, is employed for frequency conversion from RF (2-500MHz) to the 70MHz IF.

6. SOFTWARE

The software architecture of this prototype is based on SCA v2.2 as shown in Fig. 3. Relationship between the *OE* and *Application* is shown in Fig.4. Access limits to the OS service are based on SCA v2.2.

The software components selected for the embedded real-time OS and CORBA are shown in Table 2. While the SCA v 2.2 supports real-time extension of POSIX, the requirement for real time accuracy in the prototype adapter software drove the selection of the CPU reservation function of Time Sys Linux, which can specify the CPU processing time [3].

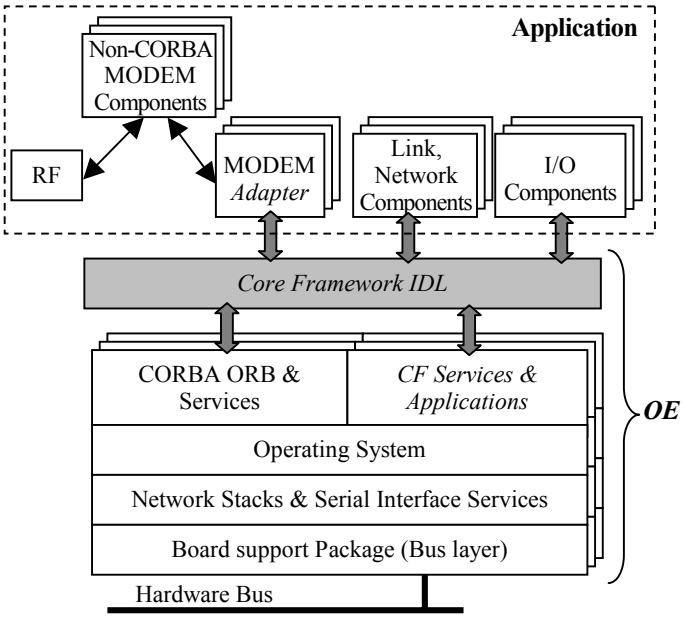

Figure 3. Software Structure

Table 2. OS and CORBA

Layer	Software
Operating System	Time Sys Linux / CPU[3]
CORBA	ORB express RT for C++ v2.5 [4]

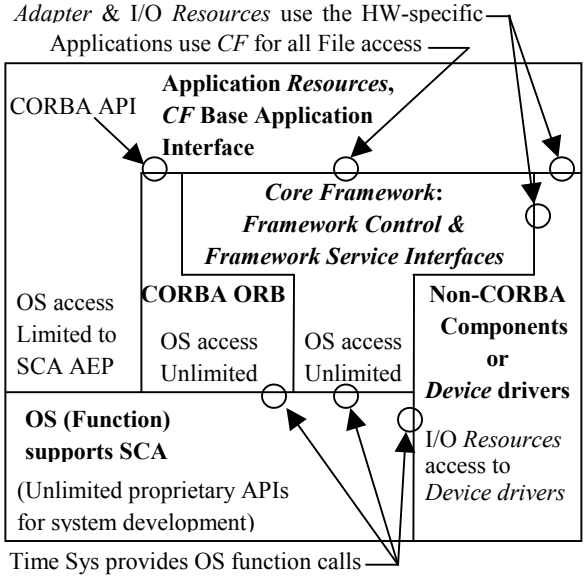

Figure 4. Relationship between OE and Application
(No *Devise Interface* is provided)

6.1 Operating Environment

The software layer of the *OE* is constructed identically to the structure layer of SCA v2.2 as shown in Fig.3.

CORBA-Naming service is arranged in the *CORBA ORB & Service* based on SCA v2.2. For Event message transfer, SCA v2.2 states to use Event channels that will be produced by *Event service* of CORBA. In the prototype, however, Log service is used for the Even*t* message transfer, transferring only to the Control terminal (see 6.3 (2)), since no Event channel is managed due to absence of the *Domain Manager* (see 6.2.1).

The software bus is formed by CORBA using PCI bus. The ORB Express software, originally designed for Ethernet [4], can be used for CORBA (see Table 2) by developing the means to connect the PCI device driver with the TCP layer

.

6.2 CF Services & Applications

6.2.1 Reservation of implementation for Device interfaces and related interfaces

Implementation of the *Interfaces* of *Device, Device Manager, and Domain Manager*, defined in the *CF Services & Applications* in SCA v2.2, are reserved in this prototype as described in a) - c) below:

a) The *Device Interface* aims at building an environment enabling *Waveform Applications* to be implemented on various hardware devices. However, it is considered that definitions in SCA v2.2 on processing of the functions called through the *Device interface* are insufficient to assure such portability. For example, when *Waveform Applications* by other manufacturer are installed in the hardware, the *Device interface* may not discriminate the type of CPU, or correctly identify other hardware

specific resources. It is necessary to provide additional detail, through extended definition of the *Domain Profile,* or use of a concise *Hardware Abstraction Layer (HAL)* API, in order to put the *Device interface* to practical use ("Waveform Portability for Software Defined Radios [5]", by GDDS in the technical conference 2002, discusses the role of the *Device interface* for portability. Also, the SDR Forum is preparing to issue a Request for Information (RFI) for a *HAL* API). In conclusion, implementation of *Device interface* is reserved in the prototype until a future time when the above issue is resolved. (*Resource Interface*, which is generalized one of *Device Interface*, is inherited).

b) The *Device Manager interface* aims at managing the utilizable *Device/Service interfaces*. Considering the absence of *Device Interface* and that *Service Interfaces* can be managed using Naming Service if necessary, implementation of *Device Manager* is reserved.

c) The *Domain Manager Interface* aims at managing utilizable *Interfaces of Device, Device Manager, Service, Application, and Event channel*. Since the *Device Manager Interface* is absent, management of the *Application Interface* will not be necessary, and Event message service can be provided by *Human Computer Interface (HCI)* with Log service, implementation of the *Domain Manager* Interface is also reserved for the future.

To cope with the above reservation of the Interfaces, the following means are provided:

1) Profiles in the *Domain Profile*, which are relevant to the absent *Interfaces of Device* and *Domain Manager*, are reserved for future implementation. Therefore, the operation behaviours, "*Device* management of the *allocate Capacity/de-allocate Capacity* included in the *Device Interface*, and *load/unload/ execute/ terminate* for software" are performed with the *HCI*. These are executed by reading out an index file where software arrangement information is described instead of *Domain Profile* relevant to the *Interfaces of Device* and *Domain Manager*.

2) The *HCI* also directly calls the creation of applications to an *Application Factory* interface instead of calling via *Domain Manager Interface*.

3) I/O component software directly accesses to a *Device* driver(s) instead of accessing via *Device Interface*.

4) Registration of *CF Services & Applications* (and *Resource Interfaces* of *waveform applications*) are performed by registering its own object reference into the CORBA naming service so that other software can be obtained object references, instead of the prescribed register operation to be contained in the *Domain Manager Interface*.

5) Instead of the means to register the *File Manager* and *File System* Interface by the *Device Manager/Domain Manager Interface* at the system start up, the *File Manager* registers its own object reference into CORBA naming service at the start up time so that the *File Manager Interface* can be used.

6.2.2 File Manager, File System, File Interface

The operation functions and attributes are implemented, using Linux functions. For the Linux file system, Journaling Flash File System (JFFS) is used for a longer life for number-of-writing times of the flash memory. Mount of each *File System* Interface is performed utilizing Linux system initialization software.

6.2.3 Application Factory, Application Interface

The *Application Factory* can build up to 32-patterns of *Waveform Application* per channel.

Although the *Application* Interface in SCA v2.2 specifies implementation of operation to be delegated to *Assembly Controller*, in the prototype, it is delegated to the implementation of operation, made by *Application Interfaces* to each software since the *Assembly Controller* is not provided.

For the XML purser engine for *Domain Profile* read out, Expat software is used. The XML purser is designed as a class module that is read out in the header file of *Application Factory* not as independent software.

6.3 Services

Log service and *Event service* are implemented in this prototype.

Log Service: The function and attribute of operation are implemented inheriting *CF* IDL of Log Interface. Like *CF Services &Applications* Interface, the Log interface registers its own object reference into CORBA naming service.

Event Service: Although there is no event message inside domain due to absence of Interfaces of *Device* and others in the prototype, there are Event messages to be informed to the domain exterior (Control terminal). These are transmitted to the exterior via the HCI that is polling to the Log by writing into *Log service*.

6.4 Design Policy not defined in SCAv2.2

The *Waveform Application* of this prototype is built by inter-connecting each component separated by function based on SCA v2.2, as shown in Fig. 5-7. The following policies are adopted though these are not specified in the SCA v2.2.

1) In the Modem component, for use of FPGAs in the Quadrature Modems, and DSPs in the Baseband Modem, *Adapter software* is distributed to the CPU in each Modem.

2) Software in the Link/Network components is distributed to CPUs of the module so as to operate on CPUs where processing loads are not concentrated.

3) The I/O components are distributed to the CPU in interface module so that they can always access to the device driver and operate on the CPU.

4) C++ development language is used throughout the components.

6.5 Application Program Interface

Although the SCA v2.2 specifies use of an *Application Program Interface (API)* described in the API Supplement [6] in order to ensure compatibility between software components by different manufacturers, a simpler service API, not complying with the Supplement, has been temporarily adopted in the prototype. This takes into account the following issues:

- As described in 6.2.1, Interfaces that aim at compatibility between manufactures are reserved for future definition.
- Further study and possible revisions will be necessary for the Supplement to assure compatibility between JTRS Cluster radios.

7. TYPICAL WAVEFORM APPLICATIONS

As shown in Fig.5-7, block diagrams of typical *Applications,* and explained 1)-3) below, protocol change is made only by switching to connection objects of the components. Components required for application composition, connection of Ports, *configure*, etc. are described by *Domain Profile.*

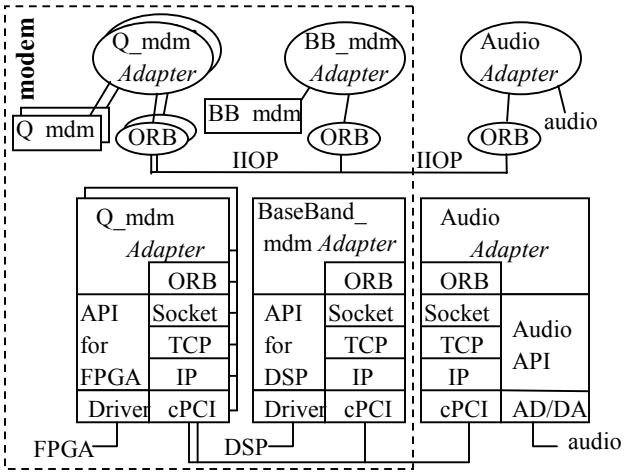

Figure 5. Voice Application Block Diagram

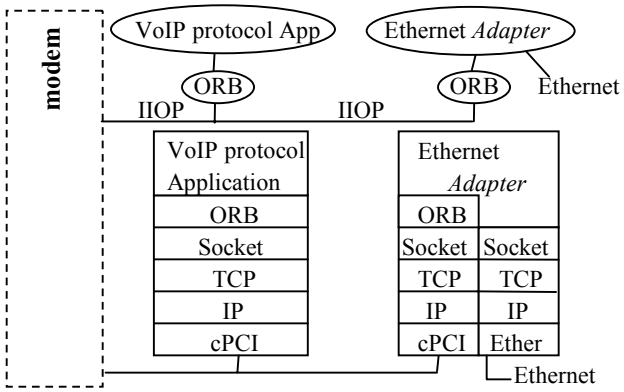

Figure 6. VoIP Application Block Diagram

1) ***Voip Application*** (Fig.6): can be made replacing the Audio adapter component in Fig.5 with a VoIP protocol application component and adding Ethernet adapter component.

2) ***Data Transmission Application*** (Fig.7): can be made replacing the VoIP protocol application component in Fig.6 with a Data transmission protocol application component. Ethernet used for connection with the Control terminal can be replaced for serial (RS232C) use, by connection change from Ethernet Adaptor to a Serial Adapter, and the communication mode change made by *configure* operation of the *property set* interface belongs to *Base Application Interfaces*.

3) ***Utility Application***: Fig.8 shows an example in which PA (the peripheral equipment) and RF unit are controlled. It never disturb *Waveform Application* operation even when they are operated simultaneously, since data volume for the *Utility Application* is quite small compared with a software bus.

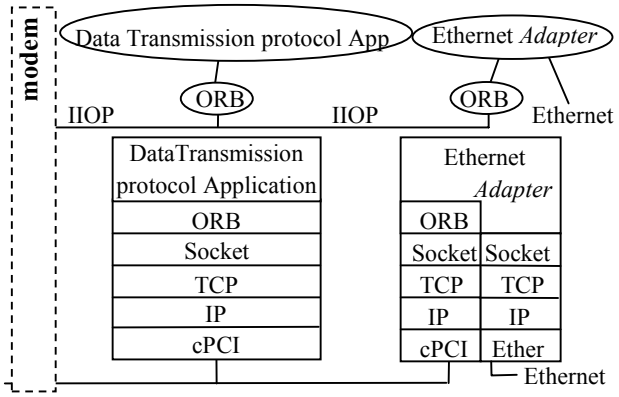

Figure 7. Data Transmission Application Block Diagram

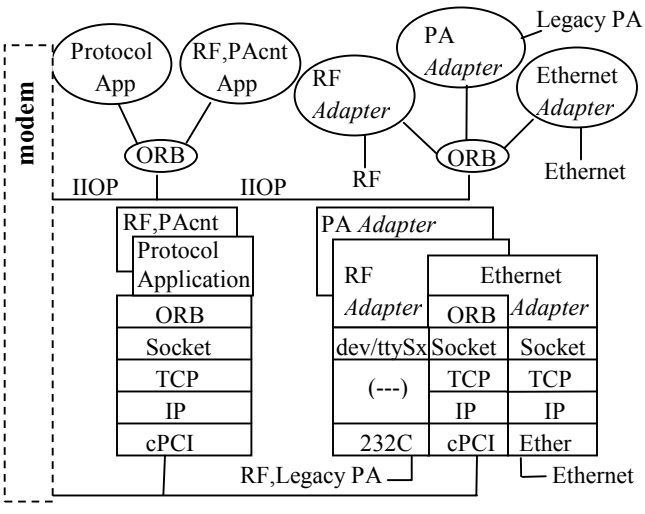

Figure 8. RF/PA Utility Application Block Diagram

8. PERFORMANCE EVALUATION

SDR of this software architecture presents excellent performances in flexibility of software that constitutes *Applications* and expandability of the *Application* functions. Alteration or addition of software necessary for various *waveform applications* is minimized, and most can be utilized. For addition of software, only creation of *Domain Profile* is necessary besides creation of additional software (Modification of *Application Factory* may be needed depending on the case). The identical features as above are presented for alteration or addition of peripheral equipment control utility applications.

It was confirmed that *Utility Application* (such as peripheral equipment control) *and Waveform Application* can operate simultaneously without mutual disturbance (It was difficult before SCA was introduced).

9. COMMENTS ON SCAV2.2

For the *waveform application* and its execution environment, further study will be necessary to examine the following points.

(1) Flexibility of *waveform application*:

1) Section 6.2.1 discussed the portability of the interfaces of *Device*, *Device Manager*, and *Domain Manager,* pertaining to "Waveform portability for SDR [5]" as presented by GDDS. Since these interfaces may not be related to the portability of software applications, we question their indispensability, particularly for a light weight SCA implementation [7].

2) In the current SCA, it is difficult to provide flexibility of device types to the software which operates on *Device*, and it is desirable for hardware vendors to offers description domain profile software and an adapter for *Device*. Moreover, since it may be difficult to judge what function *Device* has, it may be difficult for the others to use *Device,* even though *Device* is managed by *Device Manager* or *Domain Manager*. Further study is required for the above two issues.

(2) OE: The OE should specify the following items:

1) The regulation about use of a hard real-time functional library provided by OS vendor, or a unified library from multiple vendors.

2) Processors & OSs: The regulation for maintaining the compatibility of application (example: JAVA, Linux Standard Base (LSB) etc.). For example, it is better to add compiler service to the OE, since compile is needed, if distribution of source code is allowed.

3) The regulation about addition of XML purser interface to CF, or registering a purser into service of *Domain Manager*.

(3) Download: The interface for downloading is added to the CF service. The download behavior and the sequence of processing that are implemented in the interface are specified further.

(4) Architecture Compliance: In order to clarify the compliance level of radio and application, the details of the compliance level of SCA are specified.

10. CONCLUSION

Continuing to the last year's effort, evaluated hardware design approaches for multi-band, multi-mode SDR, we developed a new prototype with upgraded hardware and adoption of the SCA v2.2, examining our software approach and *waveform applications*.

Through our trials such as utilization and addition of components, our research platform demonstrates the effectiveness of an SDR based on SCA v2.2 for flexibility and expandability of *waveform applications*. It was also confirmed that *Utility Application and Waveform Application* are able to operate simultaneously without mutual disturbance. (Before SCA introduction, It was difficult to achieve)

Although the evaluation was performed for two channels configuration, extension to four channels are in preparation. For higher speed operation and efficient use of the PCI bus, two present modems of the Quadrature and Baseband will be integrated in the next model.

Based on our development experience to date, we have submitted some of our "lessons learned" about the SCA v2.2 to the SDR forum.

Our goal is production of SDR that are platform agnostic, and application compatible with future enhancements of the SCA. We are going to continue development of our SDR product.

11. REFERENCES

[1] I. Teshima,K. Takahashi,Y. Kikuchi,S. Nakamura and M. Goami, "Development of Software Radio Prototype", Proceedings of the 2002 Software Defined Radio Technical Conference, pp.169-174, November11-12, 2002.

[2] Software Communications Architecture (SCA) Specifications MSRC-5000SCAv2.2, 17 November 2001.

[3] TimeSys Web Site : http://www.timesys.com.

[4] Objective Interface Web Site : http://www.ois.com.

[5] G. Foresman, G. Osborn, D. Dohse, E. Christensen, "Waveform Portability for Software Defined Radios", Proceedings of the 2002 Software Defined Radio Technical Conference Volume 1, pp.67-73, November 11-12, 2002.

[6] Application Program Interface Supplement to the Software Communications Architecture Specification, MSRC-5000API, V1.0 December 15, 2000.

[7] D. Murotake, A. Fuchs, A. Martin, B. Fette, J. Reed, M. Robert, "A Lightweight Software Communications Architecture (SCA) Launcher Implementation for Embedded Radios" (Draft), to be presented at the 2003 Software Defined Radio Forum Technical Conference, Orlando Florida, USA, November 17-19, 2003.