APPLICATION PROGRAM INTERFACE FOR DIGITAL POLAR TRANSMITTERS

John D. Bard, Ph.D.
Space Coast Communication Systems, Inc.
Melbourne, FL, USA, jbard@spacecoastcomm.com

Walid K. M. Ahmed, Ph.D.
M/A-COM, Tyco Electronics
Morristown, NJ, USA, ahmed@tycoelectronics.com

Abstract

An Application Program Interface (API) for a radio transmitter is defined that accommodates various analog and digital configurations. In addition to up-conversion - perhaps using one or more digital or analog stages – controls affecting gain, bandwidth, and timing should be visible that support all modulation types and access protocols. Furthermore, whereas traditional radio transmitter topology maps one conversant with one transmitter, software radio can accommodate several conversations over several carriers on a single transmitter. Each carrier can have individually configured bandwidths, power levels and modulation types. With the addition of low power and small form factor requirements, conventional IQ modulators and surface acoustic wave (SAW) filters begin to run into efficiency and linearity problems that erode their performance in multimode applications.

A new digital polar design that facilitates SDR transmitters has been developed by M/A-COM, Tyco Electronics. M/A-COM's new transmitter technology features a complete, digital base-band to RF/PA transmit chain that comprises a novel Digital Power Amplifier (DPA) and Transmit IC. The wideband DPA uniquely provides RF power amplification, phase/amplitude combining and amplitude modulation in a single device. This device is complemented with a Digital Modulator to realize a compelling Digital Transmitter (DTx) architecture. The fully digital nature of the DTx allows it to efficiently adapt, through programmability, to various modulation standards, e.g., GSM/EDGE, cdma2000, over a wide range of frequencies with minimum complexity. The unprecedented multi-mode/multi-band nature of the DTx qualifies it as a strong candidate for future software defined radios, particularly for handsets.

As opposed to current implementations, which utilize an analog interface, DTx requires a digital interface to the baseband. As described, this interface accommodates multi-carrier operation. In addition to digital data, a control interface is also described. Control parameters include clock rates, modulation mode of operation (optional), carrier frequency, and filter coefficients. Additionally there should be some way of synchronizing data flow with control information. Finally, it is to be emphasized that the digital nature of M/A-COM's transmitter technology is indeed an enabler for software defined radio and all the advantages it promises.

In this paper, we introduce M/A-COM's software radio transmitter topology and propose an API set that supports a wide variety of transmitter configurations including unconventional implementations that offer reduced part count, smaller footprint and lower power consumption.

1. INTRODUCTION – THE REQUIREMENT

The definition of an Application Program Interface (API) is provided in [1]; "The interface between the application software and the application platform, across which all services are provided. The application programming interface is primarily in support of application portability". Additionally, the Systems Interface Working Group of the Software Defined Radio (SDR) Forum "feels it is necessary to provide input on the importance and development of APIs for SDR to the general community." [2]. For those working with the Department of Defense, API's are non-negotiable: "Developers are required to submit APIs for approval and Configuration Management by the JTRS JPO" [3]. So given this undeniable necessity for SDR API's, where's the beef?

2. SDR API HISTORY

The perceived need for API's, its criticality to the success of SDR and the requirement for published, non-proprietary interfaces can be traced back to meeting number two of the SDR Forum [4]. This recognition resulted in the development of the MMITS API Definition (MAD) process [5] – at least we recall our state of mind at the time. The

latest work is now occurring in the Object Management Group (OMG) with the SDR Forum being one of the technical contributors. We'll examine an API that was developed before the SCA and the SDR Forum Technical Report v 2.1, and understand its applicability towards a lower-level ubiquitous API that has so far eluded us.

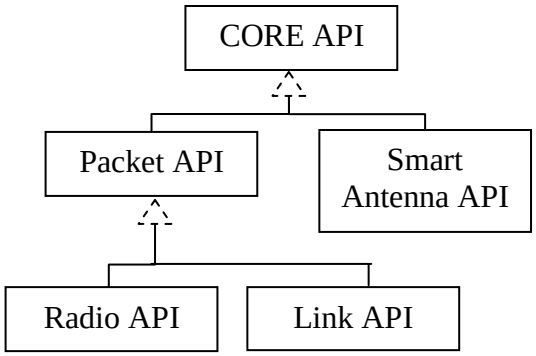

Figure 1: Framework Inheritance Model

3. THE GLOMO API

The Global Mobile Information System was a DARPA program circa late 90' that focused on developing new wireless ad hoc networking technologies. Part of the effort resulted in a Smart Antenna API and a Radio Device API [6]. Additional API specifications existed for a Framework super-class and an extension for TDMA transceivers. The reason we are looking back at the GLOMO API is that it pre-dates the SCA wherein radio devices and services were completely abstracted away. We will examine the features of the GLOMO API and then compare it against the feature set of the latest Platform Independent API's from the OMG and SDR Forum. Finally, we will suggest a further stratification of the GLOMO API's and then apply them to M/A-COM's Digital Transmitter technology.

Similar in concept to the SCA, the GLOMO API defines a Core (Framework) API from which device API's inherit. Figure 2 shows a subset of the GLOMO API's relevant to the radio transceiver. The little inheritance triangles are dashed as an indication that the implementation was actually done in "C", but that a formal inheritance mechanism should be used for languages that support inheritance. The Radio Device API includes a structure that can be replicated for each channel. Thus for a multi-channel capability the API provides for the notion of an array of Radio Device objects. The GLOMO API supports a strong separation of the Link and Media Access Control (MAC) layers – this is also consistent with the SCA. Essentially the Radio Device is unaware and furthermore doesn't care about the specific network protocols in which it is passing back and forth over-the-air.

A good Radio API will allow for the fact that certain events within the radio are asynchronous and not under direct control of the network protocols. The networking function needs to be made aware of these events but they will occur without the provocation of the controlling function. A poorer choice of API would involve a polling mechanism wherein the controller would query a particular event or set of events. The GLOMO API calls these asynchronous events "signals". Figure 2 shows the collaboration between the (Link/MAC/Network) controller and the Radio Device.

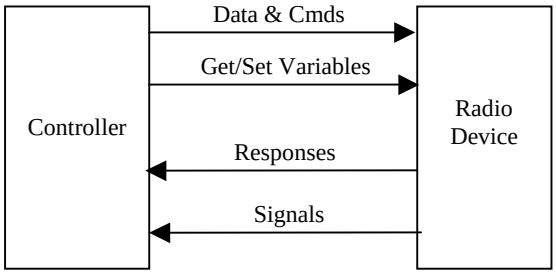

Figure 2: GLOMO Collaboration Diagram.

Essentially there are two types of commands. One is the "Get/Set Variables" which refer to persistent states or parameters associated with the channel. The Controller also gets a chance to advance or retard the state of the radio. Between Controller-issued commands and asynchronous signals from the Radio, the Controller has precise knowledge of the state of the over-the-air segment of the link. This is consistent with the SCA's requirement for an API to support non real-time control as well as real-time control and data [7]. This brings up a feature of the Packet API that was identified but not implemented - the ability to attach device specific commands to the data packet. The command set would be different in talking to a serial port device as opposed talking to a radio device. This is where some of the difficulty in its implementation would lie. You don't want to have the carry around the baggage of every different type of device in the system in every packet. It does, however represent a critical functionality for the software radio – the ability to synchronize the command and data streams.

Finally, before delving into the detail of the GLOMO Radio Device API and comparing it to the latest work of the OMG one should consider the relationship between the abstraction of the Radio Device Object and its implementation. In GLOMO there was a separate Radio Physical Interface API. That essentially described a lower level driver style API complete with interrupt lines and

timing diagrams. Thus Figure 1 can be re-drawn to include the Radio Physical API.

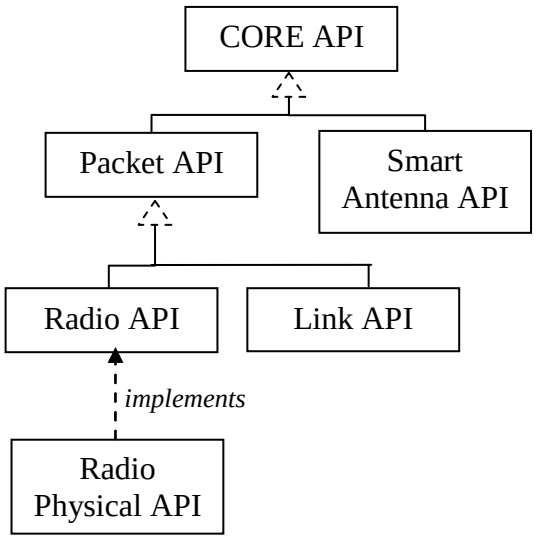

Figure 3 – GLOMO API Diagram

This is exactly the kind of implementation detail talked about in the last sentence of paragraph 2.1 of the JTRS JPO API Standardization Process [3]

In accordance with the portability and technology insertion objectives of JTRS ORD, abstract API's shall be fully defined in a platform independent language such as the Unified Modeling Language (UML). In order to accommodate the various Cluster implementations, concrete API's shall be defined in their platform specific or native language definitions.

4. COMPARISON OF THE GLOMO API AND THE OMG API

The comparison we're about to attempt is quite awkward due to the huge difference in the level of abstraction between GLOMO and the OMG's Software Radio Platform Independent Model [8]. Given the limited page count of this medium, our comparison will not be an exhaustive one.
The OMG's view of the software radio is that it is a set of services upon which applications and management functions execute. The software radio services themselves are offered as facilities: Common, Data Link, IO, Physical and Radio Control. First, we examine the Common Layer facilities that are available to all data streams to see if it supports two elements critical to software radios (especially legacy JTRS applications) - that is flow control and the ability to pass control data along with the data. Flow control – supports watermarking, under-run/over-run and ACK/NAK. (GLOMO API supports only over-run). Control information is bundled with data in packets called Protocol Data Units or PDU's. A PDU is the smallest size packet that has meaning in any waveform layer. Each packet contains a ControlHeaderType, a Platform Specific Model – CORBA IDL is offered for the control header:

```
struct ControlHeaderType {
   AddressType sourceAddress;
   AddressType destinationAddress;
   long priority;
   SduSizeType sduSize;
   long sequenceNumber; };
```

This control header is combined with a data octet sequence to form a PDU. There is no field (or fields) in the control header to support the passing of device specific commands. How does one synchronize the certain commanded events within a facility to a particular packet of data? Page 255 of the Software Radio Components PIM/PSM offers the following statement "Real-time control and signals are communicated via the packet interface." Yet such control and signaling, that is, beyond the ControlHeaderType, is not defined.

Finally, we delve down into the modem and RF facilities to compare with the GLOMO Radio API. The modem facility includes the necessary functionality to implement the over-the-air protocol, AM, PSK, QAM, CPM, etc. Additionally the modem offers interfaces such as inter-leaver, forward error correction, PN sequence generation, etc. The interaction between these elemental blocks is configurable at instantiation.

A key feature of the GLOMO API is the ability of the Radio Device to asynchronously signal the controlling entity that an event has occurred. Of course in the OMG specification a complete event service is identified [9]. Briefly the event service is used as follows. For simplicity, two actors are considered, 1) the modem or event supplier and 2) a man-machine interface - event consumer - to display the event. The event itself is inconsequential but say for instance is the detection of a carrier sync pattern by the modem. Administratively, something (in our example the modem object itself) creates an event channel. The modem retrieves a reference to the supplier-side channel and the consumer-side channel. The modem then obtains a "push" proxy to the consumer and connects to that proxy. When the event occurs a simple call to push on the consumer proxy will send the signal to the MMI. Incidentally that signal itself can contain a payload of type "any" which unfortunately has a large overhead associated with it. So love it – or use something else – the PIM/PSM does make provision for the modem facility (or any other facility) to signal asynchronous events.

The purpose of the RF/IF facility is to adapt the symbol stream to the transmission channel by adjusting the frequency response, power, and centre frequency of the

signal. This has no corollary in GLOMO because the modem and RF functionality was combined. The RF/IF facilities offer interfaces for devices such as: amplifier, switch, frequency converter, hopping frequency converter, antenna and digital converter. Interestingly enough, each component in the RF/IF chain has an inherited interface called Frequency Response. There are two configurable attributes on Frequency Response and that is the tuned (or center) frequency and the gain/phase response relative to center frequency. In the elemental case a single point can specify a cutoff frequency. For example, a 100 kHz first order low pass response is configured as *tunedFrequency* = 0 Hertz, *FrequencyResponsePoint*[1] = {frequency = 100,000 Hertz, amplitude = -3 dB, and phase = -45 Degrees}. More complicated responses can be constructed as an array of *FrequencyResponsePoints* relative to a *tunedFrequency*.

5. CLOSER TO THE IMPLEMENTATION

It is clear that the functionality and architecture of the GLOMO API maps nicely underneath the CORBA-based Software Radio PSM. The combination of the Modem and RF/IF facilities offer the ability to extend the configurability offered by GLOMO. No matter how the modem and RF/IF functionality is packaged, underneath the covers there is most likely "C" code running on a DSP coupled with other functionalities executing in FPGA's and/or ASIC's. This, of course, is modeled as Platform Specific.

Though we descend into the more concrete aspects of the radio implementation this is no reason to abandon a layered design methodology. Immediate benefits exist for reduced development cost in the form of higher re-use, a design that's easier to understand and debug [10]. The ease of debug comes with the ability to be able to better isolate a defect to a particular layer whilst the layers above and below operate correctly. Figure 3 from [11] suggests such a layered approach to the embedded portion of the software radio.

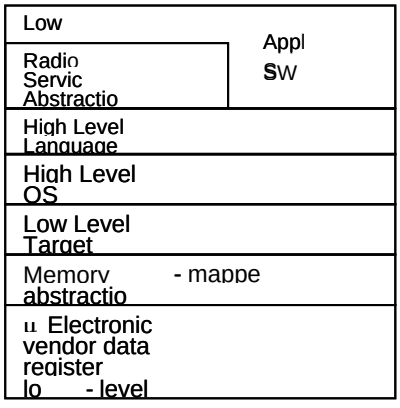

Figure 3 Abstraction Layers Underneath a PSM.

Given a selected chip set upon which to fabricate our software radio, it is better at this point to begin with the vendor data sheets and design ever-increasing levels of abstraction leading out to the Software Radio Platform Specific model.

6. THE DIGITAL TRANSMITTER

A new digital polar design that facilitates SDR transmitters has been developed by M/A-COM, Tyco Electronics. M/A-COM's new transmitter technology features a complete, digital base-band to RF/PA transmit chain that comprises a novel Digital Power Amplifier (DPA) and Transmit IC. The wideband DPA uniquely provides RF power amplification, phase/amplitude combining and amplitude modulation in a single device. This device is complemented with a Digital Modulator to realize a compelling Digital Transmitter (DTx) architecture. The fully digital nature of the DTx allows it to efficiently adapt, through programmability, to various modulation standards, e.g., GSM/EDGE, cdma2000, over a wide range of frequencies with minimum complexity. The unprecedented multi-mode/multi-band nature of the DTx qualifies it as a strong candidate for future software defined radios, particularly for handsets.

7. DIGITAL TRANSMITTER ARCHITECTURE

Figure 4 depicts a high level abstraction of the Digital Transmitter architecture, which consists of two modules, namely, the Digital Modulator (DM) module and the Digital Power Amplifier (DPA) module.

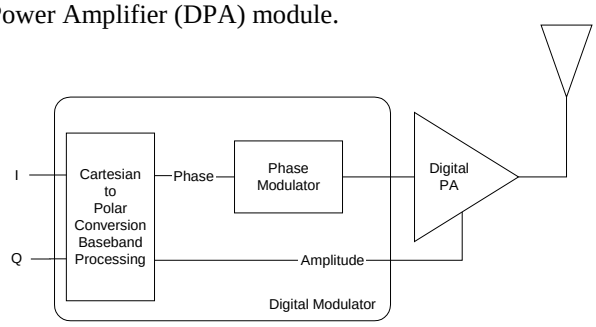

Figure 4: The DTx Digital Polar Modulator Block Diagram

The DM module converts the native digital baseband I/Q signals from the Cartesian domain to the polar domain. This digital interface eliminates the need for baseband D/A converters and reconstruction filters. This block also performs the signal processing to meet spectral mask requirements and compensate for AM/AM and AM/PM distortions. The phase information is passed through a phase modulator, yielding an on-channel, phase-modulated carrier. The phase-modulated carrier is fed into the Digital Power Amplifier (DPA), along with the amplitude modulation information. The two signals are combined to generate a fully- modulated carrier, with the required output power

signal level. The combining of the magnitude and phase signals takes place at the final output stage of the DPA, allowing the earlier stages of the DPA to operate in compression. . This digital transmitter approach can also be re-configured for multi-band/multi-mode operation: The DTx is tunable through an off-channel synthesizer to support different bands. There is no band-specific hardware, and therefore the DTx can be configured for different frequency bands and modulation schemes by simply re-configuring clock frequencies and filtering coefficients to support the desired standards. Finally, power control; an important issue in the design of CDMA transmitters, is accomplished with multiple VGA stages operating in the transmit chain. Because the DTx is a direct conversion, polar-based approach, it follows that all the gain control must take place at RF frequencies.

8. CHARACTERISTICS OF THE DPA

When used in conjunction with the Digital Transmitter, the Digital Power Amplifier offers several advantages:

- *The DPA is capable of wideband amplitude modulation*. Amplitude bandwidths associated with all the major modulation schemes are readily accommodated (envelope bandwidths of 10MHz and beyond).
- *Performing amplitude modulation at the last stage of the DPA gives reduced current drain over the transmit power control range*. The final stage of the DPA is biased into Class-B or Class-C operation. Additionally, the driver stages are operated non-linearly for efficiency, consuming very low quiescent current (<30mA).
- *The DPA facilitates efficient power control at all levels of RF power*. A dynamic range of 55dB is realized under CDMA modulation. This is possible because the gain is applied to a constant-envelope waveform that contains only phase information. The phase is insensitive to distortion and, consequently, the DPA stages are biased to operate non-linearly for efficiency over the power control range. Additional power control dynamic range can be obtained through the phase path stages, to satisfy the CDMA requirement of ~ 80dB of power control dynamic range.

9. DIGITAL BASEBAND-TO-RF INTERFACE CONSIDERATIONS

I/Q Data Interface Considerations

Since the DTx operates in a fully-digital mode, it is most efficient when it is supplied with digital I/Q data in addition to the control signals associated with the operation of the DTx. Various scenarios could be thought of, for example, the DTx can be supplied with only the symbol-level I/Q representation. Then, pulse-shaping can be done within the DTx, provided that the associated filter coefficients, sampling rates, … etc. are supplied on the control signals. The digital interface can be serial or parallel. Providing the symbol-level I/Q data is probably the most suitable form for SDR, since symbol-level signals require the lowest possible sampling-rate. In addition, deciding on parameters such as the I/Q symbol bit-width becomes performance-independent, since the symbol-level bit-width is directly and exactly defined by the modulation scheme rather than by the implementation quality of the pulse-shaping filters, which are implementation-dependent.

Control Signals Interface Considerations

The control signal interface would carry information such as pulse-shaping filter coefficients, clock-rates, sampling rates, power amplifier AM/AM/PM correction tables, power-control-related information, … etc. The control signals interface can be parallel or serial.

Considerations for Multi-Carrier Operation

In order to accommodate multi-carrier operation, one can think of several architectures that can make use of the DTx digital nature. The “brute-force” architecture is to build one digital modulator (DM) module and multiple DPAs for the multiple carriers that need to be transmitted. The DM module can then be configured/programmed to support the multi-carrier operation and will be equipped with multiple output interfaces to provide the multiple amplitude signals and their corresponding phase-modulated RF carriers to the DPAs. Clearly, an RF mechanism is needed in order to couple/superimpose all the modulated carriers onto one antenna (or array of antennas).

Finally we offer a somewhat generic interface that has the feature of allowing a control message to be passed along with a data packet. The following IDL code fragment offers such an interface:

```
module MACOM_Physical_DTx {

   struct DTxControlPhys {

      /* Control Structure */
      CF::OctetSequence DTx_control;
   };

   interface TransmitPacketPhys {

   /*  This operation is used to push
       Client data to the Server with a
       Control element and a Payload
       element. */

      oneway void pushPacket (
            in DTxControlPhys control,
            in CF::OctetSequence payload
      );

   };
};
```

REFERENCES

[1] NSA Cross Organization CAPI Team, "Security Service API: Cryptographic API Recommendation" Object Management Group, 95-06-06, 12 June 1995.

[2] Systems Interface Working Group, "API Position Paper", , SDR Forum SDRF-03-A-0005-V0.00, July 19,2003.

[3] S. A. MacLaird, Col., USAF, "Application Program Interface (API) Policy", Department of the Army JTRS Policy 002, June 24,2003.

[4] B. Fette, "Speakeasy Phase II", SDR (MMITS) Forum, June 11,12, 1996.

[5] P. Cook, "The MMITS API Definition Process", SDR (MMITS) Forum, March 17, 1998.

[6] D. Beyer, et. al, "Radio Device API", Roof Top Communications, July 11 1998, http://www.ir.bbn.com/projects/udaan/udann-index.html

[7] API Supplement, v1.0, Figure 3-2, JTRS Joint Program Office, November 17, 2001.

[8] M. Bicer, G. Bickle, et al, PIM and PSM for Software Radio Components – Final Adopted Specification, Object Management Group, dtc/04-05-04.

[9] Object Management Group, Event Service Specification version 1.1, formal/2001-03-01.

[10] S. Finseth, "Abstracting Device-Driver Development", Embedded Systems Programming, May 2004, pp. 32-36.

[11] J. Bard, "Device-Centric SDR Solutions and the Software Communications Architecture", Government Micro-Circuit Applications and Critical Technology Conference (GOMAC), Monterey, CA, 15-18 March.