CONTROL AND TEST SOFTWARE FOR IRAM WIDEX CORRELATOR

S. Blanchet, D. Broguiere, P. Chavatte, F. Morel, A. Perrigouard, M. Torres
IRAM, Grenoble, France

Abstract

IRAM[1] is an international research institute for radio astronomy. It has designed a new correlator called WideX for the Plateau de Bure interferometer (an array of six 15-meter telescopes) in the French Alps. The device started its official service in February 2010. This correlator must be driven in real-time at 32 Hz for sending parameters and for data acquisition. With 3.67 million channels, distributed over 1 792 dedicated chips, that produce a 1.87 Gbits/sec data output rate, the data acquisition and processing and also the automatic hardware-failure detection are big challenges for the software. This article presents the software that has been developed to drive and test the correlator. In particular it presents an innovative usage of a high-speed optical link, initially developed for the CERN ALICE experiment, associated with real-time Linux (RTAI) to achieve our goals.

INTRODUCTION

After the installation of its new generation of receivers in 2006, IRAM has started to build a new correlator called WideX that can process up to 64 GHz of total analog bandwith [1]. It is a big machine with 3.67 million channels, distributed over 1 792 custom[2] chips. With such a number of components, powerful programs for automatic testing are crucial, otherwise the project fails. Real-time driving at 32 Hz and data processing is also a serious software challenge because the total device output data rate is about 1.87 Gbit/s.

TECHNICAL CHOICES

Data Transmission

For technical convenience the correlator is divided into four identical sub-units (Fig. 1), driven by four computers. It means that it requires 4 data links at 448 Mbit/s. Finding a way to transfer data at such rate, was a headache for engineers. Fortunately CERN had already developed such a high-speed serial link for its own needs[3] and it was possible to buy this technology jewel at a very affordable price. The CERN solution is called Detector Data Link (DDL) [2]. It is a versatile high-speed optical link, that can transfer up to 1.5 Gbit/s. It is composed of:

[1]Institut de Radioastronomie Millimétrique

[2]The correlator chip is a 250 MHz application-specific integrated circuit (ASIC) designed under IRAM specifications [1].

[3]CERN has developped DDL for the LHC/ALICE experiment.

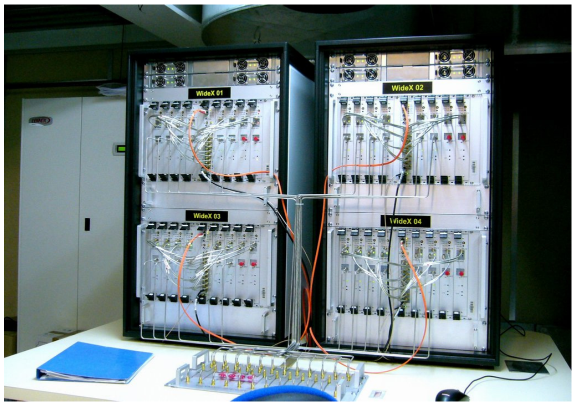


Figure 1: IRAM WideX on the observatory site. Note the four orange fibers that connect the four sub-units.

- a *Source Interface Unit* (SIU) to plug into the correlator (Fig. 2).

Figure 2: CERN Source Interface Unit board.

- a *Destination Interface Unit* (DIU) hosted on a read-out receiver card (Fig. 3) to connect to a computer.

Figure 3: CERN Read-Out Receiver Card (PCI-X slot) with 2 Destination Interface Units.

- a fiber optic to link directly the SIU and the DIU (orange fibers on Fig. 1).
- a Linux driver with the source code.

This optical link is very easy to use: the data are sent to the SIU through a parallel bus, then they appear automatically in the computer memory at the address that has been provided to the driver.

Real-time Operating System

While processing large amounts of data, the computer must drive the data acquisition at 32 Hz, therefore a real-time operating system is required. Linux-RTAI[4] was chosen because it has already been successfully used at IRAM, for critical tasks like the antenna control.

This real-time engine guarantees that the data acquisition runs at 32Hz, without losing any data, whatever the computer load is. Among the different RTAI running modes, LXRT[5] was prefered, because it allows hard real-time programs to run in user space: the software development is easier and programs can be written in C++ instead of C only.

The only small drawback is that the CERN driver and libraries are not directly compatible with RTAI, but a modification of the CERN source code has solved this problem.

Software Libraries

Because electronic engineers need very fast and responsive testing tools, the fastest software libraries have been carefully selected to build the programs:

- Qt: *a free C++ cross-platform framework* [4]. It provides fast and nice graphical widgets and also many other very useful classes. It is developped by Nokia Corporation.
- Qwt: *Qt Widgets for Technical Application* [5]. It is a Qt extension, that provides nice technical widgets like a plotter (QwtPlot) with built-in zoom and autoscaling. It is a community project.
- FFTW: *Fastest Fourier Transform in the West* [6]. It is a free library to compute the Discrete Fourier Transform in one or more dimensions. It uses code-generation and runtime self-optimization techniques to achieve a very high level of performance. This software library was developed at the Massachusetts Institute of Technology (MIT).

SOFTWARE

The different pieces of hardware impose some constraints on the time diagram (Fig. 4) that should be explained to understand the most interesting details of the software design.

1. The correlator data output is scheduled on an internal 32 Hz clock[6].

[4]RTAI: RealTime Application Interface [3]. It is a real-time extension for the Linux kernel from the *Department of Aerospace Engineering, Politecnico di Milano, Italy*.

[5]LXRT: Linux Extension for Real Time. It is a special RTAI scheduler.

[6]The 32 Hz frequency is dictated by the Walsh functions that are used to reduce the effects of electrical crosstalk between antenna signals.

2. To read data, the computer sends a `RDYRX` (Ready-to-Receive) command before the 32 Hz pulse. The transfer starts with the 32 Hz pulse.
3. The data transmission is long and uninterruptible. Therefore it allows only a small sending window after `EOBTR` (End Of Block Transfer).
4. The CERN board generates no hardware interrupts: a polling is required to detect the end of transmission (`EOBTR`).
5. The CERN board writes data directly in memory without using the central processor, therefore it is the best moment to process data from the previous acquisition period.

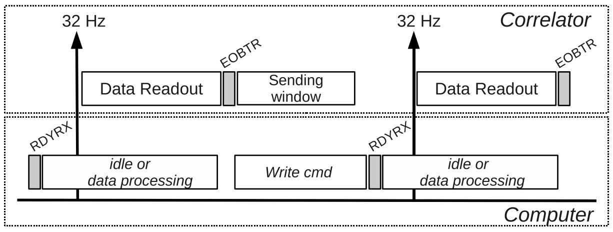


Figure 4: Time diagram.

Time Synchronization

Like the antenna position, the correlator acquisition parameters change relentlessly with the Earth rotation. Unlike the correlator that synchronizes itself directly on the time signal of the observatory maser, the computer has no particular hardware for clock synchronization. However a software-only solution works very well:

1. Because the transmission duration is constant and starts always on a 32 Hz pulse (Fig. 4), the computer considers the end-of-block-transfer (`EOBTR`) event as a faithful image of the correlator clock. Therefore the real-time tasks are synchronized on this event. The polling loop starts only when required, at the very last time, and exits after only few iterations. In this case, it makes sense to use a high-frequency[7] loop.
2. To know the absolute time associated to each `EOBTR` event, the computer uses the Network Time Protocol[8] server that depends on the observatory maser.

Software Architecture

There is only one real-time program: `CONTROL` (Fig. 5). It has two real-time threads: `DRIVE` and `PROCESS`.

[7]In our case, the polling loop runs at 1 kHz, but it could be higher.

[8]Network Time Protocol (NTP) is a protocol for synchronizing the clocks of computers over variable-latency networks.

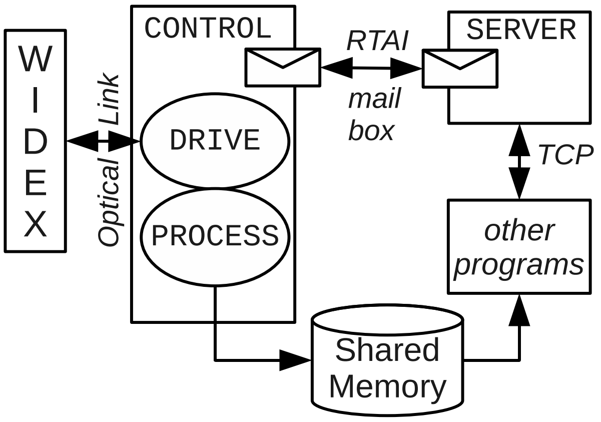


Figure 5: Software architecture.

- DRIVE drives the data acquisition. It has the first real-time priority to avoid any data loss. It executes the following loop of instructions:
  1. Get a free memory page[9], and send its address to the CERN driver.
  2. Send a RDYRX command to the correlator, and then sleep[10] until just before the next EOBTR event.
  3. Poll until EOBTR occurs.
  4. Push the page of the just-arrived data into the waiting queue of the PROCESS thread,
  5. Send any pending parameters to the correlator.
  6. Sleep until just before the next 32 Hz pulse.
  7. Repeat the loop.
- PROCESS processes the raw data from the correlator. It has the second real-time priority to guarantee that the processing is finished in the expected time. It executes the following loop of instructions:
  1. Take the first data page from its waiting queue.
  2. Process the data and write the result into the shared memory.
  3. Push back the memory page to the pool.
  4. Repeat the loop.

Because of the chosen priorities, the Linux kernel and all the other software run only when DRIVE and PROCESS are asleep. To send specific command to the correlator, programs must pass through SERVER (Fig. 5) to transform TCP requests into RTAI messages. The command will be really sent to the correlator during the next sending window (Fig. 4).

[9]The memory page comes from *physmem*: it is a reserved memory area for CERN DDL direct memory transfers. A realtime-compatible memory allocator has been specifically written for this area because the Linux kernel does not manage it.

[10]The sleeping and the transmission last exactly the same time, but the sleeping ends before, because it starts before.

RUNNING MODES

There are different running modes, one for each testing stage.

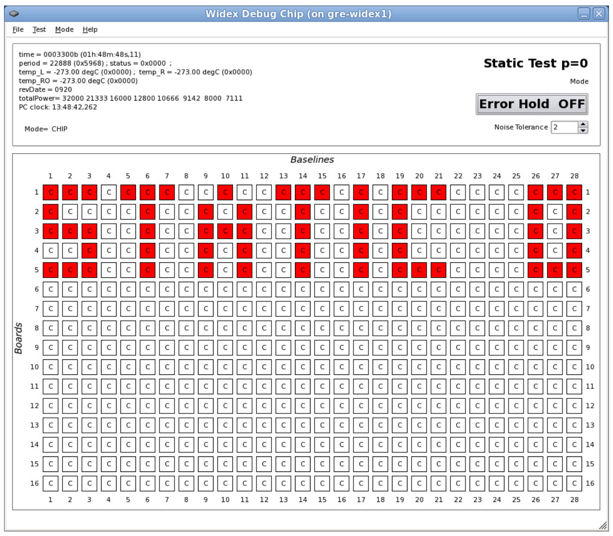
Simulation Mode

The target of this mode is to test software without needing hardware, and also to prove that the processing algorithms are correct. The simulation mode changes the behaviour of the driving task: instead of reading the CERN driver output, the thread calls real-time functions that generate custom simulation data.

There are several simulation patterns, one per kind of tests. For example, Fig. 6 shows a simulation pattern for chip failure detection.

Chip Debugging Mode

The goal of this mode is to check that all the correlator chips work as expected. This mode modifies the processing task: instead of analyzing spectrum, it compares together the raw outputs of the chips. A special program displays the results as a 448-cells matrix (Fig. 6). When a problem is detected in a chip, the cell color changes. The user can click on the chip to display its channel values (Fig. 7).



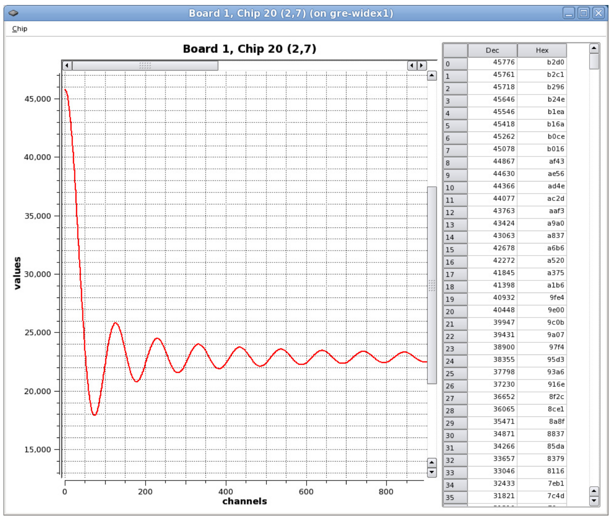
```
time = 0003300b (01h:48m:48s,11)
period = 22888 (0x5968) ; status = 0x0000 ;
temp_L = -273.00 degC (0x0000) ; temp_R = -273.00 degC (0x0000)
temp_RO = -273.00 degC (0x0000)
revDate = 0920
totalPower= 32000 21333 16000 12800 10666  9142  8000  7111
PC clock: 13:48:42,262

Mode= CHIP
```

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	
1	🟥C	🟥C	🟥C	C	🟥C	🟥C	🟥C	C	C	🟥C	C	C	🟥C	🟥C	🟥C	C	🟥C	C	🟥C	🟥C	🟥C	C	C	C	C	🟥C	🟥C	🟥C	1
2	🟥C	C	C	C	C	🟥C	C	C	🟥C	C	🟥C	C	C	🟥C	C	C	🟥C	C	🟥C	C	C	C	C	C	C	🟥C	C	🟥C	2
3	🟥C	🟥C	🟥C	C	C	🟥C	C	C	🟥C	🟥C	🟥C	C	C	🟥C	C	C	🟥C	C	🟥C	C	C	C	C	C	C	🟥C	C	🟥C	3
4	C	C	🟥C	C	C	🟥C	C	C	🟥C	C	🟥C	C	C	🟥C	C	C	🟥C	C	🟥C	C	C	C	C	C	C	🟥C	C	🟥C	4
5	🟥C	🟥C	🟥C	C	C	🟥C	C	C	🟥C	C	🟥C	C	C	🟥C	C	C	🟥C	C	🟥C	🟥C	🟥C	C	C	C	C	🟥C	🟥C	🟥C	5
6	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	6
7	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	7
8	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	8
9	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	9
10	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	10
11	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	11
12	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	12
13	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	13
14	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	14
15	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	15
16	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	16
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	

Figure 6: Chip debugging window. A simulation pattern is printing messages on the chip matrix.

Observing Mode

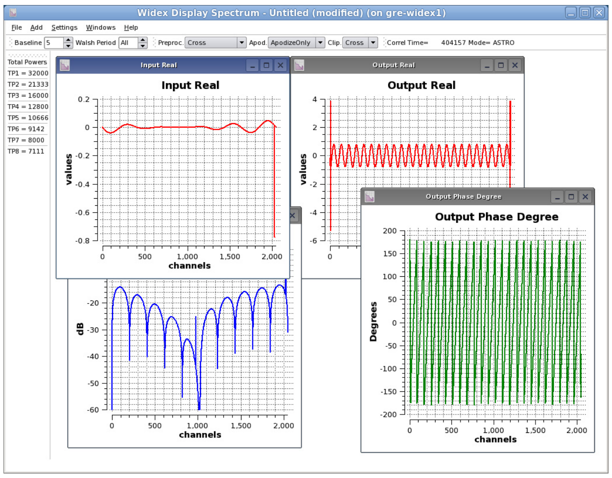
The purpose of this mode is to show that the correlator can extract the hidden signal from the noise. Therefore it implements almost all the operations for real observing. This mode modifies the processing task to analyze the spectrum in real-time. These signal processing operations are organized in sequence: Fast Fourier Transform followed by several corrections of levels and phases. Each pipeline stage can be enabled or disabled in order to measure its



	Dec	Hex
0	45776	b2d0
1	45761	b2c1
2	45718	b296
3	45646	b24e
4	45546	b1ea
5	45418	b16a
6	45262	b0ce
7	45078	b016
8	44867	af43
9	44630	ae56
10	44366	ad4e
11	44077	ac2d
12	43763	aaf3
13	43424	a9a0
14	43063	a837
15	42678	a6b6
16	42272	a520
17	41845	a375
18	41398	a1b6
19	40932	9fe4
20	40448	9e00
21	39947	9c0b
22	39431	9a07
23	38900	97f4
24	38355	95d3
25	37798	93a6
26	37230	916e
27	36652	8f2c
28	36065	8ce1
29	35471	8a8f
30	34871	8837
31	34266	85da
32	33657	8379
33	33046	8116
34	32433	7eb1
35	31821	7c4d

Figure 7: Content of a chip. The curve is plotted with Qwt-Plot [5].

individual effect and its contribution to the total execution time. A fast viewer (Fig. 8), with direct buttons to control signal processing, has been written to display the results. It heavily uses Qt [4], Qwt [5], and FFTW [6].


Widex Display Spectrum - Untitled (modified) (on gre-widex1)
File Add Settings Windows Help
Input Real
values
channels
Output Real
values
Output Phase Degree
Degrees
channels
dB
channels

Baseline 5 Walsh Period All Preproc. Cross Apod. ApodizeOnly Clip. Cross Correl Time= 404157 Mode= ASTRO

Total Powers
TP1 = 32000
TP2 = 21333
TP3 = 16000
TP4 = 12800
TP5 = 10666
TP6 = 9142
TP7 = 8000
TP8 = 7111

Figure 8: Dedicated viewer for correlator spectrum.

CONCLUSION

Thank to the innovative pair CERN DDL – RTAI, it is possible to build a very high-performance control software. It drives in real-time a complex scientific device, while processing large amounts of data. The icing on the cake is that the solution is built with free open-source software only.

ACKNOWLEDGMENT

The authors would like to thank Ervin Dénes, Csaba Soós and Pierre Vande Vyvre from CERN: we are grateful for their free hardware loan and their precious help at the early stage of the project.

REFERENCES

[1] M. Torres, "Main Technical features of the WideX correlator" `http://www.iram.fr/widex`

[2] CERN DDL website `http://cern.ch/ddl`

[3] RTAI website `http://www.rtai.org`

[4] Qt website `http://qt.nokia.com`

[5] Qwt website `http://qwt.sourceforge.net`

[6] FFTW website `http://www.fftw.org`