ADVANCED USES OF THE WORLDFIP FIELDBUS FOR DIVERSE COMMUNICATIONS APPLICATIONS WITHIN THE LHC POWER CONVERTER CONTROL SYSTEM

Q. King[1]
[1]CERN, Geneva, Switzerland

ABSTRACT

The LHC power converter control system is based on an embedded computer designed at CERN called a Function Generator/Controller (FGC). Every one of the 1700 LHC power converters will include an FGC responsible for the control and monitoring of the converter's state (on, off, standby, etc…) as well as the generation of the current reference and the regulation of the measured current. The operation of the LHC will require the coordination of all 1700 power converters, so synchronisation is fundamental. Given the large scale of the system, both in terms of the number of nodes and their geographical spread (27 km), it was decided to use a fieldbus for communications. It was found that by choosing the WorldFIP standard, which has a real-time behaviour, it was possible to support data communications and synchronisation over the same twisted pair cable.

This paper describes how the underlying WorldFIP protocol has been adapted to support all the communications requirements of the FGC. These include: (i) a packeting layer for commands and responses, (ii) 50 Hz publication of power converter and circuit status, (iii) broadcast of UTC time and events (e.g. start ramp, end ramp), (iv) 50 Hz transmission of real-time reference values to support orbit, tune or chromaticity feedback, (v) broadcast of software updates, and (vi) support for remote terminal access. Of particular note is the use of the WorldFIP fieldbus to discipline a software phase-locked loop (PLL) that enables the FGC's real-time clock period to be aligned to UTC to within 1 part in 10^7. This requires no special hardware as the FGC's Motorola HC16 microcontroller (like most microcontrollers) includes an on-chip general-purpose timer. The paper shows how to implement a generic PLL in software and provides a web-link to a test program that can be downloaded to experiment with the PLL algorithm.

It is difficult to overstate the significance of the use of a single fieldbus for these diverse communications applications. It has enabled a huge reduction in cabling and connectors (and therefore cost) when compared to previous controller designs in which timing and communications required independent networks, and we can anticipate an associated improvement in reliability.

INTRODUCTION

The new Large Hadron Collider (LHC) will reuse much of the civil engineering infrastructure of its predecessor, LEP. However, for various reasons, the LHC will use many more power converters than LEP (~1700 instead of ~800), though their geographical spread around the 27 km of the machine will be similar.

The power converter control system for LEP worked very successfully for the lifetime of the machine and was based on individual embedded control computers linked to gateway machines by MIL1553 fieldbuses. It was decided very early on in the LHC project to use the same architecture for the LHC power converter controls, since the constraints are similar. However, MIL1553 was a very old standard by then, so a survey was made of newer fieldbuses that identified WorldFIP as especially interesting for LHC.

Key features of all fieldbuses include low cabling and component costs and high noise immunity. In general they operate at rather modest speeds over long cable lengths. WorldFIP has all these features, and one further characteristic that made it unique at the time: real-time synchronisation.

It was known from the start that the Function Generator/Controllers (FGCs) for the LHC power converters would need to support real-time control for many of the circuits, so that key beam parameters could be regulated using feedback on beam measurements. This could potentially require a dedicated real-time network, which would add significantly to the cost and complexity of the system. Furthermore, independent of any real-time requirements, all the FGCs will need to operate synchronously, so distribution of timing and events is mandatory. In LEP, this was done using a separate timing network in parallel to the MIL1553 fieldbuses.

The developers of WorldFIP decided to trade bandwidth for determinism when they designed the WorldFIP protocol. For example, when running the physical layer at 2.5 Mbps, the fieldbus can transmit synchronisation with a jitter of less than 5 µs, and data throughput of about 1 Mbps. This is a cost effective exchange as it enables one network infrastructure to distribute synchronised time, timing events, real-time control and status data, software updates and commands and responses.

OVERVIEW OF THE WORLDFIP FIELDBUS

The only way to make a network deterministic is to rigorously control the traffic, and the WorldFIP protocol does this by assigning special responsibilities to one node, the so called "bus arbitrator" (BA). No other nodes transmit data unless invited to do so by the BA node. The BA follows instructions in a table that define which transmissions are to be solicited at each moment within each period of the WorldFIP macro cycle. Once the end of the table is reached, the last instruction can either tell the BA to restart immediately, resulting in a free-running network, or to wait for an external synchronisation pulse before starting again.

All nodes, including the node that is acting as the BA, can be so called "producers" and/or "consumers" of WorldFIP data, which is packaged either as "variables" or "messages" (see below). The BA table can schedule "periodic" variables or messages to be sent repetitively at given moments within the macro cycle. The table can also schedule "aperiodic" windows for variables and messages. Aperiodic transmissions are made on request and those that cannot be satisfied during one cycle are not forgotten by the BA and carry over the next cycle. In this way, real-time periodic data can be guaranteed to be sent when required, while non-real-time aperiodic data will be sent as soon as possible. Elaborate BA tables can be created containing multiple elementary cycles that schedule different real-time variables and/or messages at different rates, while preserving a certain fraction of the bandwidth for aperiodic traffic.

Physical layer

Two companies have developed silicon for the WorldFIP fieldbus; Alstom and Schneider. Their chipsets target slightly different applications and are not compatible. Alstom's chipset is the best adapted for use by independent developers such as CERN and includes two types of interface chip: FullFIP and MicroFIP. The MicroFIP chip is a low-cost variant that can be a data producer/consumer, but cannot be a bus arbitrator (BA). It has a very simple interface for the software which makes it perfect for embedded control systems. By contrast, the FullFIP device can be a BA and has an extremely complicated software interface that requires powerful computing hardware to run.

The WorldFIP physical layer is relatively standard for a fieldbus, and uses the same 150 twisted pair cabling as Profibus from Siemens. Up to 32 nodes can be coupled to a bus segment by transformers (this also provides galvanic isolation). The maximum cable length per segment depends upon the bus speed: 500 m at 2.5 Mbps, 750 m at 1 Mbps and 1900 m at 31.25 kbps. Up to three repeaters can be included to link segments together if longer distances or more nodes must be accommodated. Optical fibre transceivers can be used if groups of nodes are widely separated. Overall, the WorldFIP physical layer is very robust electrically and mechanically and measured error rates are very low.

WorldFIP Variables

A WorldFIP variable has the following characteristics: (i) a fixed length of up to 64 bytes (when using the MicroFIP device), (ii) a 16-bit numeric identifier, (iii) one producer and any number of consumers. One device will be configured as the producer of a given variable, and it will transmit it whenever requested to do so by the BA. Other nodes can be configured to consume the variable if required, so WorldFIP variables can effectively be node-to-node, multicast or broadcast. The producer does not need to know which other nodes, if any, will make use of the variable.

WorldFIP messages

A WorldFIP message has the following characteristics: (i) a variable length of up to 128 bytes (when using the MicroFIP device), (ii) a six byte header that includes the destination and source addresses, (iii) an optional acknowledgement system that can confirm transmission. The destination address can either be a specific node or a special broadcast address received by all nodes.

USING WORLDFIP FOR THE LHC POWER CONVERTERS

Six equipment groups at CERN have chosen to use WorldFIP for their communications. Three selected the 1 Mbps variant, two will use 31.25 Kbps and for the power converter controls we chose 2.5 Mbps and accepted that in some cases we will need repeaters when 500 m is insufficient. For bandwidth reasons, we limited the number of FGCs per segment to 30 and chose a macro cycle period of 20 ms. This is the fastest rate that is compatible with our required combination of the real-time and non-real-time traffic. Figure 2 shows the architecture of a typical FGC WorldFIP network. There will be about 70 networks around the LHC linking the ~1700 FGCs to the central LHC controls network.

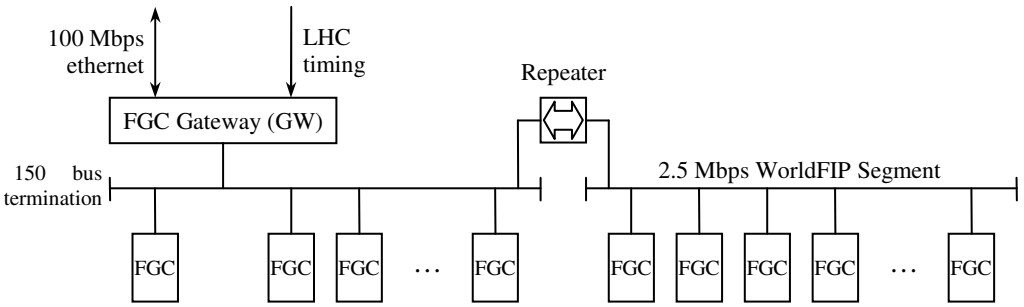

Figure 1: Architecture of a typical FGC WorldFIP network

The macro cycle encoded in the BA table in each gateway is illustrated in figure 2. Periodic variables produced by the gateway are indicated by boxes above the line (T, A, B) and those produced by the FGCs are indicated below the line (1, 2, 3, …, 29, 30). The grey zone shows the window set aside each cycle for aperiodic messages.

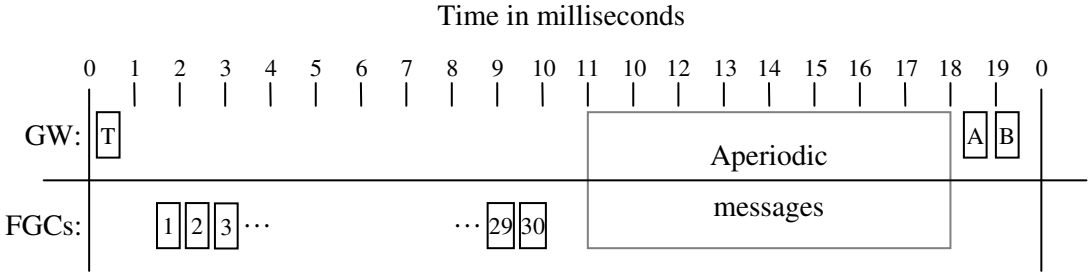

Figure 2: WorldFIP macro cycle for the FGC segments

Packeting layer for commands and responses

The traditional requirement for any distributed control system is to be able to send commands to individual devices and to receive responses. With WorldFIP, aperiodic messages are the natural solution for this requirement; however, the maximum message payload of 122 bytes is not sufficient for long commands or responses. Thus we use two bytes as a header, which identifies whether the message is a first, last, or in-between packet. Furthermore, the FGCs acknowledge the reception of every command packet they receive by toggling a bit in the status variable that they produce on the next cycle. It therefore takes two macro cycles per packet, giving a maximum throughput from the gateway to an FGC of 3000 bytes/s. This appears rather slow, but the gateway can send up to eight packets per cycle, so it can communicate at this rate with 16 FGCs simultaneously.

Responses to commands are not individually acknowledged by the gateway; however, the header includes an incrementing sequence number that enables the gateway to know if a packet went missing. Thus an FGC can send up to 6000 bytes per second to the gateway. The aperiodic message window during each macro cycle is long enough to allow about ten full length messages to be sent. This bandwidth is shared between the gateway and FGCs communication needs in a first-come first-served

manner, so if lots of FGCs attempt to respond at the same time, each will see a reduction in their throughput to the gateway.

The gateway will detect packets that go missing in both directions, but this is so rare that it will not attempt to retransmit the command. Instead, it will simply report a communications error to the client application that sent the command, which must then decide what to do.

Real-time publication of converter and circuit status

Every FGC will be requested by the BA to transmit a 56 byte periodic WorldFIP variable containing status information (1 to 30 on figure 2). This 50 Hz real-time channel provides an essential link between the FGCs and the outside world. The 56 bytes are divided into many sections: (i) command packeting protocol support, (ii) power converter and FGC diagnostics, (iii) faults, warnings and other digital status information, (iv) state machines and (v) analogue signals. The gateway uses some of the data itself, while other parts drive the interfaces to the LHC alarm system, logging, post-mortem, diagnostic interfaces and status web pages. The data is also available for subscription by applications.

Broadcast of synchronised time and events

The start of each macro cycle is synchronised by a 50 Hz timing signal from a timing receiver interface in the gateway. The jitter in the timing signal is less than 1 µs, and the WorldFIP interface typically adds 2 µs (the rare worst case is 6 µs) when starting the first transmission of the cycle. This is always the 64-byte periodic variable produced by the gateway and consumed by every FGC (T in figure 2). The MicroFIP interface in each FGC generates an interrupt request (IRQ) when this variable is received. This IRQ provides the synchronisation pulse for the software phased-locked loop that disciplines the local real-time clock in each FGC (see the next section). The contents of the variable include the time of day (UTC encoded as UNIX time) and the time of the next timing event (START_RAMP or ABORT_RAMP) if one is pending. The time variable is also used for the distribution of software updates as described below.

Real-time transmission of current reference

A predefined current reference for a given circuit can be specified as a table that the FGC will interpolate to derive the current reference as a function of time $I(t)$. For some circuits it will be necessary to send a value in real-time, so that feedback can be used to regulate beam parameters such as orbit or tune. For simplicity, the WorldFIP network sends an individual 32-bit floating point value to *every* FGC *every* cycle (50 Hz), whether needed or not. Within each FGC the real-time value can then be: (i) ignored $(I_{ref} = I(t))$ (ii) additive $(I_{ref} = I(t) + \ _{rt})$ (iii) multiplicative $(I_{ref} = I(t).(1 + g_{rt}))$, or (iv) direct $(I_{ref} = I_{rt})$, according to the setting of a property called MODE.RT.

It would be inefficient to send individual messages or variables to each FGC, so instead, the real-time values are bundled into two WorldFIP variables (A and B in figure 2). Two variables are needed because the maximum variable length is limited by the MicroFIP interface to 64-bytes and 120 bytes are required to send thirty floats. We tried using one periodic message containing all the floats but it was unreliable; the FullFIP interface would sometimes stop sending the messages for no apparent reason. Instead the real-time values for FGCs 1-15 are sent in variable A and those for FGCs 16-30 are sent in variable B. Each FGC knows its number and programs its MicroFIP to only consume the appropriate variable. When the variable is received, it then only reads the four bytes that belong to it from the interface. This solution is very efficient for the FGCs and works reliably.

Each real-time value takes effect at the start of the next cycle so the variables A and B are sent at the end of each cycle to reduce the transmission latency. The transmission deadline will be the start of millisecond 18, so real-time data from controls applications must reach the gateway by then (using UDP over the Ethernet), if they are to be available in the FGC on the next cycle.

Broadcast of software updates

The FGC software is constantly evolving, and this is expected to continue even beyond the start-up of the LHC. In LEP, the power converter controller software was stored in EPROMs, so changing the software was very time consuming and was done only twice during the lifetime of the machine. By contrast, the FGCs use flash memories to store their software, so reprogramming is easy either from the WorldFIP network or from the FGC's serial interface. The approach we have taken was inspired

by satellite TV decoders, which can reprogram themselves from a continuously cycling data stream included with the TV signals broadcast by the satellite. The FGC gateways broadcast the 64-byte WorldFIP variable T every cycle, of which 44 bytes are dedicated to code distribution (12 bytes are header, 32 bytes are payload). In this way, 1600 code bytes can be broadcast per second.

The FGC software and databases are broken into different "Codes". These are sent continuously in a cycle that takes less than ten minutes. Eight bytes of every T variable provide one row of what is called the "advertised code table". This table has ten rows and therefore takes 200 ms to be received. After a reset, an FGC uses this table to know if it should wait for a new code or can start operating immediately. This method means that FGCs always update themselves when required, and updating all 1700 systems in the LHC will take no longer than updating a single system.

Remote terminal support

One very positive experience from LEP was the use of an ASCII command protocol that allowed interaction with a controller using a dumb VT100 compatible terminal. The FGCs maintain this principle and local control is always possible via the RS232 interface. One very valuable feature of the WorldFIP connection is the ability to project this terminal interface remotely. This is achieved by using two bytes of the T variable to send keyboard characters; one byte is the target node's ID, the other is the keyboard character. Characters to be displayed on the screen are returned in a special section of the FGC's response messages. In the FGC, remote keyboard characters are merged with local keyboard characters so any number of remote terminals and the local terminal can be used simultaneously. This makes it very easy for an expert to provide support remotely to teams installing and commissioning power converters in the LHC.

LOCAL REAL-TIME CLOCK AND SOFTWARE PHASE LOCKED-LOOP

Synchronisation is of fundamental importance to the power converter controls in the LHC. When ramping the main circuits at 10 A/s, an error in time of just 1 ms corresponds to an error in current of nearly 1 ppm. If communications are lost, the FGCs should be able to operate autonomously with minimum temporal drift compared to the LHC reference time source (GPS atomic clocks). For a drift of less than 1 ms over 1000 s, the frequency error must be less than 1 ppm. This is possible if a local real-time clock is accurately disciplined by a phase-locked loop (PLL). Traditionally, this has required a dedicated analogue oscillator circuit, but it is possible to implement a PLL in software using only the on-chip timer of the microcontroller. To do this requires two elements: (i) a real-time clock with fine period control, (ii) a PLL to regulate the clock period.

Real-time clock with fine period control

Like most microcontrollers, the FGC's HC16 has an on-chip general purpose timer (GPT) which has a free running fast clock and registers to capture the time of input signals (input capture), and registers to generate timing pulses and/or interrupts (output compare). The fast clock is derived from the microcontroller's clock, which is driven by a crystal oscillator with good stability at a constant temperature, but an absolute error of up to 100 ppm. A trivial implementation of a real-time slow clock is shown in figure 3.

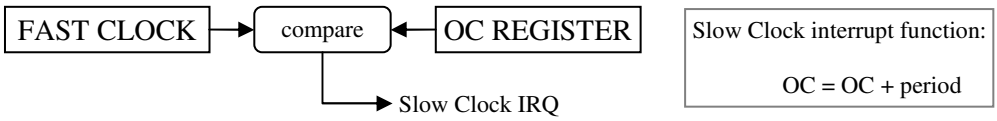

Figure 3: Trivial implementation of a slow clock using an Output Compare (OC) register

The limitation of this implementation is the resolution of the period. If the fast clock runs at ~1 MHz, and the slow clock should be 1 kHz, then the period will be ~1000, but since the period is an integer, the period of the slow clock can only be adjusted in steps of 0.1%, or 1000 ppm. To provide finer control of the period we need a fractional part. If we can accept a jitter in the slow clock of up to one fast clock period, we can then use the fractional part of the period to make the frequency precise when averaged over many cycles. This is trivial computationally because of the behaviour of unsigned integer additions. Since the HC16 fast clock and OC register are both 16-bit, we can make

the period 32-bits with the low 16-bits as the fractional part (0-65535) and the high 16-bits as the integer part (nominally 1000). The size of the fractional part defines the averaging period to be 65536 times the slow clock period (1 ms) = ~65.5 s. A fractional part of 1 would add one fast clock period per averaging period, so the average frequency can be regulated in steps of 1 µs per 65.5s, or ~0.015 ppm. We can benefit from this finer resolution by adding a 16-bit unsigned variable to act as the OC fractional counter in the slow clock interrupt function:

```
oc_fraction = oc_fraction + fractional_period
OC = OC + integer_period + carry
```

(carry bit is set when oc_fraction *rolls over)*

For example, if the slow clock period is 1000:00001, the carry bit will be set once per 65536 slow clock cycles, and thus one extra fast clock period will be added to the slow clock per averaging period. If the period is 999:65535, then the carry will be set every cycle *except* once per 65536 and one fast clock period will be subtracted per averaging period.

Phase-locked loop

Having set up a way to finely control the average period of the slow clock, it is then possible to discipline this period using a software phase-locked loop based on a proportional-integral (PI) controller. The period of the synchronisation pulses must be a multiple of the slow clock period and they must trigger the capture of the fast clock time in an input capture register, and trigger an interrupt.

Figure 4: Using an Input Capture (IC) register to capture the time of the WorldFIP IRQ

The simplest implementation will lock the FIP synch pulses to the nearest slow clock edge using the following algorithm in the FIP interrupt function (ignoring conversions from 16 to 32 bit integers):

```
error = REFERENCE_PHASE + IC – OC
integerator = integrator + error
slow_clock_period = integrator + PROP_GAIN * error
```

In the FGC, the fast clock runs at 2 MHz, the slow clock at 1 kHz and the synch pulses at 50 Hz. We found that a proportional gain of 256 provides good PLL behaviour and is easy to implement in assembler. If the synch pulses must lock to a specific slow clock edge, then the algorithm needs to be modified to include limits on the error to avoid excessive clock slew rates.

Measurements have shown that this simple algorithm aligns the local clock and the synchronisation clock to better than 0.1 ppm. See http://cern.ch/info-sw-pll for more information about software (and VHDL) PLL implementations, including programs that you can download and test.

SUMMARY AND CONCLUSION

Everything described in this paper is fully tested and has been working for many months. Production of all 2000 FGCs will be complete in early 2006 and more than 100 have already been installed.

The choice of WorldFIP as a standard fieldbus for the LHC was made in part because of its unique real-time behaviour. This choice has been particularly profitable for the power converter control system since a separate timing distribution network could be eliminated, and six diverse communications needs met with the same networking infrastructure.

REFERENCES

[1] P. Alvarez et al. “PLL usage in the General Machine Timing System for the LHC,” ICALEPCS 2003, Gyeongju, Korea, October 2003

[2] J. Serrano, st al. “Nanosecond level UTC timing generation and stamping in the CERN’s LHC” ICALEPCS 2003, Gyeongju, Korea, October 2003