Factory Communications: On the Configuration of the WorldFIP Bus Arbitrator Table

Eduardo TOVAR*
Department of Computer Science
School of Engineering
Polytechnic Institute of Porto
Rua de São Tomé, 4200 Porto, Portugal
Tel.: +351.2.834052**9**, Fax: +351.2.8**3**21159
E-mail: emt@dei.isep.ipp.pt

Francisco VASQUES
Department of Mechanical Engineering
School of Engineering
University of Porto
Rua dos Bragas, 4099 Porto Codex,
Portugal
E-mail: vasques@fe.up.pt

* Author for Correspondence

Preferred Topic: Factory Communications (Fieldbusses)

Factory Communications: On the Configuration of the WorldFIP Bus Arbitrator Table

Eduardo TOVAR*
Department of Computer Science
School of Engineering
Polytechnic Institute of Porto
Rua de São Tomé, 4200 Porto, Portugal
Tel.: +351.2.834052**9**, Fax: +351.2.8**3**21159
E-mail: emt@dei.isep.ipp.pt

Francisco VASQUES
Department of Mechanical Engineering
School of Engineering
University of Porto
Rua dos Bragas, 4099 Porto Codex,
Portugal
E-mail: vasques@fe.up.pt

Abstract: The WorldFIP protocol is based on a centralised medium access control mechanism, where a specific station, the bus arbitrator (BA), controls all data transfers between different stations. At configuration time, the BA is given a list of process variables to scan along with their corresponding periods. This piece of information is known as the bus arbitrator table.

In this paper we provide a comprehensive study on how to configure a WorldFIP bus arbitrator table (BAT), in order to guarantee that periodic data transfers are performed before their deadlines. The main contributions of this paper are the provision of both an algorithmic approach for setting the WorldFIP BAT and a feasibility test to check *a priori* the timeliness requirements of the periodic data transfers.

Keywords: Factory communications, Fieldbus networks, WorldFIP, Real-time communications, Rate monotonic scheduling

Factory Communications: On the Configuration of the WorldFIP Bus Arbitrator Table

Eduardo TOVAR*
Department of Computer Science
School of Engineering
Polytechnic Institute of Porto
Rua de São Tomé, 4200 Porto, Portugal
Tel.: +351.2.834052**9**, Fax: +351.2.8**3**21159
e-mail: emt@dei.isep.ipp.pt

Francisco VASQUES
Department of Mechanical Engineering
School of Engineering
University of Porto
Rua dos Bragas, 4099 Porto Codex,
Portugal
e-mail: vasques@fe.up.pt

Abstract: The WorldFIP protocol is based on a centralised medium access control mechanism, where a specific station, the bus arbitrator (BA), controls all data transfers between different stations. At configuration time, the BA is given a list of process variables to scan along with their corresponding periods. This piece of information is known as the bus arbitrator table.

In this paper we provide a comprehensive study on how to configure a WorldFIP bus arbitrator table (BAT), in order to guarantee that periodic data transfers are performed before their deadlines. The main contributions of this paper are the provision of both an algorithmic approach for setting the WorldFIP BAT and a feasibility test to check *a priori* the timeliness requirements of the periodic data transfers.

Keywords: Factory communications, Fieldbus networks, WorldFIP, Real-time communications, Rate monotonic scheduling

1. Introduction

Local area networks (LANs) are becoming increasingly popular in industrial computer-controlled systems. LANs allow field devices like sensors, actuators and controllers to be interconnected at low cost, using less wiring and requiring less maintenance than point-to-point connections [1]. Besides the economical aspects, the use of LANs is also reinforced by the increased decentralisation of control and

measurement tasks, as well as by the increased use of intelligent microprocessor-controlled devices.

Broadcast LANs aimed at the interconnection of sensors, actuators and controllers are commonly known as fieldbus networks. In the past, the fieldbus scope was dominated by vendor specific solutions, which were mostly restricted to specific application areas. Moreover, concepts behind each proposed network were highly dependent on the manufacturer of the automation system, each one with different technical implementations and also claiming to fulfil different application requirements, or the same requirements with different technical solutions [2]. More recently, standardised fieldbuses supporting the open system concept, thus vendor independent, started to be commonly used. Particular relevance must be given to the European Standard EN 50170 [3], which encompasses three widely used fieldbuses: P-NET [4], PROFIBUS [5] and WorldFIP [6].

In this paper we address the ability of WorldFIP to cope with the real-time requirements of distributed computer-controlled systems (DCCS). In essence, by timing requirements we mean that traffic must be sent and received within a bounded interval, otherwise a timing fault is said to occur. More specifically, we provide a comprehensive study on how to configure the WorldFIP bus arbitrator table (BAT), in order to guarantee that periodic data transfers are performed before their deadlines.

The remaining of the paper is organised as follows. In section 2 we describe the main mechanisms of the WorldFIP protocol, concerning data transfer exchanges and the BAT main characteristics. In section 3, we provide both an algorithmic approach for building the WorldFIP BAT and a feasibility test to check *a priori* the timeliness requirements of the periodic data transfers. Finally, in section 4 we draw some conclusions.

2. A Brief Description of WorldFIP

A WorldFIP network interconnects stations with two types of functionalities: bus arbitration and production/consumption functions. At any given instant, only one station can perform the function of active bus arbitration. Hence, in WorldFIP, the

medium access control (MAC) is centralised, and performed by the active bus arbitrator (BA).

WorldFIP supports two basic types of transmission services: *exchanges of identified variables* and *exchanges of messages*. In this paper we address WorldFIP networks supporting only exchanges of identified variables, since they are the basis of WorldFIP real-time services. The exchange of messages, which is used to support manufacturing message services (MMS) [7], is out of the scope of this paper.

2.1. Concept of Producer/Distributor/Consumer

In WorldFIP, the exchange of identified variables services are based on a producer/distributor/consumer (PDC) model, which relates producers and consumers within the distributed system. In this model, for each process variable there is one, and only one producer, and several consumers. For instance, consider the variable associated with a process sensor. The station that provides the variable value will act as the variable producer and its value will be provided to all the consumers of the variable (e.g., the station that acts as process controller for that process variable or the station that is responsible for building an historical data base).

In order to manage transactions associated with a single variable, a unique identifier is associated with each variable. The WorldFIP data link layer (DLL)[1] is made up of a set of produced and consumed buffers, which can be locally accessed (through application layer (AL) services) or remotely accessed (through network services).

The AL provides two basic services to access the DLL buffers: `L_PUT.req`, to write a value in a local produced buffer, and `L_GET.req` to obtain a value from the local consumed buffer. None of these services generate activity on the bus.

Produced and consumed buffers can be also remotely accessed through a network transfer (service also known as *buffer transfer*). The bus arbitrator broadcasts a question frame `ID_DAT`, which includes the identifier of a specific variable. The DLL

[1] WorldFIP protocol is based on a three layered architecture: physical layer, data link layer and application layer.

of the station that has the corresponding produced buffer responds with the value of the variable using a response frame *RP_DAT*. The DLL of the station that contains the produced buffer then sends an indication of transmission of the value to the AL (*L_SENT.ind*). The DLL of the station(s) that has the consumed buffers accepts the value contained in the *RP_DAT*, overwriting the previous value and notifying the local AL with a *L_RECEIVED.ind*.

Figure 1 illustrates the case of a station with one produced buffer (for identifier *k*) and one consumed buffer (for identifier *x*). The first one can be locally written trough a *L_PUT.req* (overwriting the old value) or through a buffer transfer, in which its value is made available to other stations. The second one can be either locally read, through a *L_GET.req* service or remotely written through a buffer transfer, being its value overwritten with the new value transferred from a remotely produced buffer.

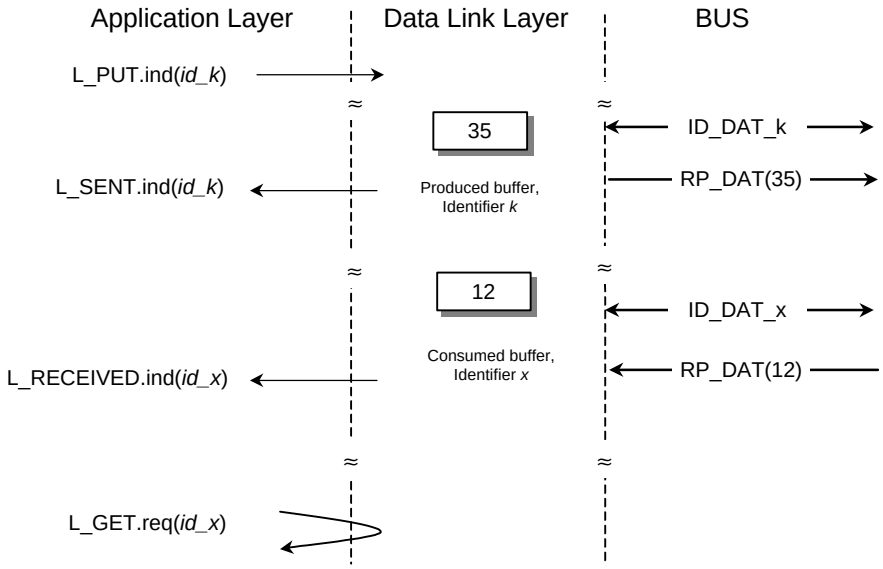

Figure 1

2.2. Buffer Transfer Timings

A buffer transfer implies the transmission of a pair of frames: *ID_DAT*, followed by a *RP_DAT*. We denote this sequence as an *elementary transaction*. The duration of this transaction equals the time needed to transmit the *ID_DAT* frame, plus the time needed to transmit the *RP_DAT* frame, plus twice the turnaround time (t_r). The turnaround time is the time elapsed between any two consecutive frames. Figure 2 illustrates the concept of an elementary transaction in WorldFIP.

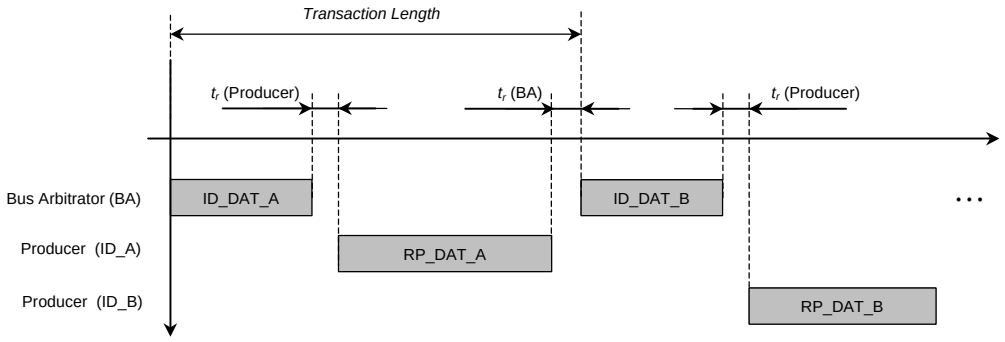

Figure 2

Every transmitted frame is encapsulated with control information from the physical layer. Specifically, the frame is placed between a DTR field (begin of frame) and an FTR field (end of frame). All WorldFIP frames begin with a control byte, which is used by network stations to recognise the type of frame, and end with two FCS (Frame Check Sequence) bytes, used by the frame receiver to verify the integrity of the received frame. The structure of both *ID_DAT* and *RP_DAT* frames is as shown in figure 3.

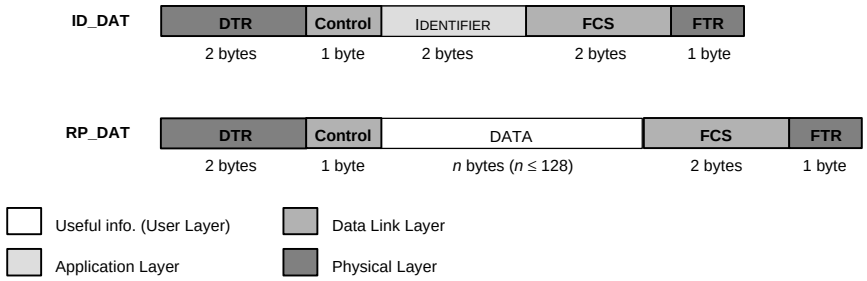

Figure 3

As it can be depicted from figure 3, an *ID_DAT* frame has always 64 bits, whereas a *RP_DAT* frame has at least 48 bits. The length of a message transaction is:

$$C = \frac{len(\text{id\_dat}) + len(\text{rp\_dat})}{bps} + 2 \times t_r \qquad \textbf{(1)}$$

where *bps* stands for the network data rate[1] and *len*(<*frame*>) is the length, in bits, of frame <*frame*>.

For instance, assuming that all variables have a data field with 4 bytes (all *RP_DAT* have 92 bits), if t_r = 20μs[2] and the network data rate is 2.5Mbps then, the duration of an elementary transaction will be (64+80)/2.5+2×20=97.6μs (equation (1)).

2.3. Bus Arbitrator Table

In WorldFIP networks, the *bus arbitrator table* (BAT) regulates the scheduling of all buffer transfers. In practice, two types of buffer transfers can be considered: periodic and aperiodic (sporadic). The BAT imposes the timings of the periodic buffer transfers, and also regulates the aperiodic buffer transfers (not addressed in this paper).

Assume a distributed system within which 6 variables are to be periodically scanned, with scan frequencies as shown in table 1. The WorldFIP BAT must be set in order to cope with these timing requirements.

Table 1: Example Set of Periodic Buffer Transfers

Identifier	**A**	**B**	**C**	**D**	**E**	**F**
Periodicity (ms)	1	2	3	4	4	6

Two important parameters are associated with a WorldFIP BAT: the *microcycle* (elementary cycle) and the *macrocycle*. The microcycle imposes the maximum rate at which the BA performs a set of scans. Usually, the microcycle is set equal to *highest common factor* (HCF) of the required scan periodicities. Using this rule, and for the example shown in table 1, the value for the microcycle is set to 1ms. A possible schedule for all the periodic scans can be as illustrated in figure 4, where we consider *C*=97,6μs for each elementary transaction.

[2] The turnaround time (t_r) is imposed [8] to be within the interval $10\times(1/bps) \le t_r \le 70\times(1/bps)$, where *bps* is the network transmission speed (in bits per second).

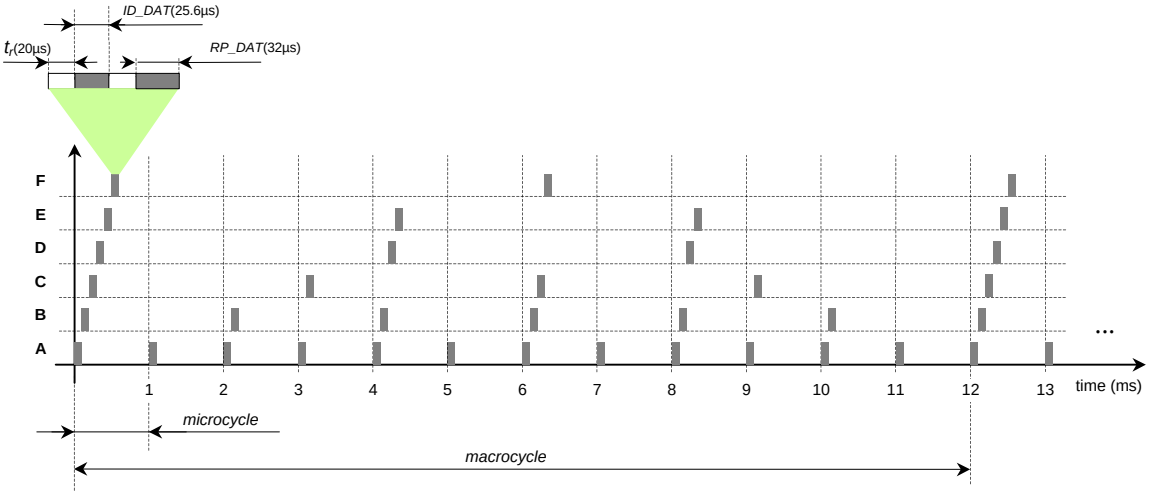

Figure 4

It is easy to depict that, for this example, the sequence of microcycles repeats each 12 microcycles. This sequence of microcycles is said to be a macrocycle, and its length is given by the *lowest common multiple* (LCM) of the scan periodicities. Thus, the setting of the WorldFIP BAT for the given example is as follows:

```
----------------------------------------
- BAT Parameters
----------------------------------------
<length of microcycle> = 1ms;
<length of macrocycle> = 12;
<identifiers in each of the 12 microcycles>
        <microcycle 1>: A, B, C, D, E, F;
        <microcycle 2>: A;
        <microcycle 3>: A, B;
        <microcycle 4>: A, C;
        <microcycle 5>: A, B, D, E;
        <microcycle 6>: A;
        <microcycle 7>: A, B, C, F;
        <microcycle 8>: A;
        <microcycle 9>: A, B, D, E;
        <microcycle 10>: A, C;
        <microcycle 11>: A, B;
        <microcycle 12>: A;
----------------------------------------
```

The HCF/LCM approach for building a WorldFIP BAT has the following properties:

1. The scanning periods of the variables are multiples of the microcycle;
2. The variables are not scanned at exactly regular intervals. For the given example, only variables *A* and *B* are scanned exactly in the same "slot"[3]

[3] For simplification of the analysis, we are assuming always the same length for each elementary transaction.

within the microcycle. All other variables suffer from a slight communication jitter. For instance, concerning variable *F*, the interval between microcycles 1 and 7 is (1-5×0.0976)+5+(3×0.0976)=5.8048ms, whereas the interval between microcycles 7 and 13 is (13×0.0976)+5+ +(5×0.0976)=6.1952ms.

3. The length of the macrocycle can induce a memory size problem, since the table parameters must be stored in the BA. For instance, if the scanning periodicities of variables *E* and *F* were, respectively, 5ms and 7ms, the length of the macrocycle would be 420 microcycles instead of only 12.

Both the communication jitter and memory size problems have been addressed in the literature. In [9], the authors discuss different methodologies for reducing the BAT size, without penalising the communication jitter problem. The idea is very simple, and it basically consists on reducing some of the scan periodicities in order to have a harmonic pattern. The problem of table size has also been addressed in other works [10,11], however, in a different perspective. In the referred work, the authors discuss an online scheduler (instead of storing the schedule in the BA's memory), which is not directly applicable to the WorldFIP case.

It is also worth mentioning that the schedule shown in figure 4 represents a macrocycle composed of synchronous microcycles, that is, for the specific example, each microcycle starts exactly 1ms after the previous one. Within a microcycle, the spare time between the end of the last scan for a periodic variable and the end of the microcycle can be used by the BA to process aperiodic requests for buffer transfers, message transfers and padding identifiers[4]. A WorldFIP BA can also manage asynchronous microcycles, not transmit padding identifiers at the end of the microcycle. In such case, a new microcycle starts as soon as the periodic traffic is performed and there are no pending aperiodic buffer transfers or message transfers. Initial periodicities are not respected, since identifiers may be more frequently scanned.

---

[4] If after the periodic traffic, the sporadic traffic (if any) and message transfers (if any) there is still time within the microcycle, the BA transmits padding identifiers, to indicate to the other stations that it is still functioning.

3. Real-Time Guarantees for the Periodic Traffic

For the periodic traffic, end-to-end communication deadlines can guaranteed by an *a priori* test, since the BAT implements a static schedule of the periodic variables. In this section we provide both an algorithmic approach for setting the WorldFIP BAT and a feasibility test to check *a priori* the timeliness requirements of the periodic data transfers

3.1. Model for the Periodic Buffer Transfers

Assume a system with *np* periodic variables (Vp_i, i = 1, .., *np*). Each periodic variable Vp_i is characterised as:

$$Vp_i = (Tp_i, Cp_i) \quad \textbf{(2)}$$

where Tp_i corresponds to the periodicity of Vp_i (assume a multiple of 1ms) and Cp_i is the length of the transaction corresponding the buffer transfer of Vp_i (as given by equation (1)).

3.2. Tools for Setting the Bus Arbitrator Table

Following the HCF/LCM methodology to build the BAT, the value for the microcycle (μCy) is chosen as:

$$\mu Cy = \underset{i=1,..,np}{HCF}(Tp_i) \quad \textbf{(3)}$$

where HCF stands for the highest common factor and corresponds to the following value:

$$\mu Cy = \max\{\Omega\} \wedge \Omega \in \aleph, \quad \text{with } \frac{Tp_i}{\Omega} = \left\lfloor \frac{Tp_i}{\Omega} \right\rfloor, \forall_i \quad \textbf{(4)}$$

An algorithm for the evaluation of the microcycle value may be as follows:

```
------------------------------------------------------------------------
- Evaluation of the Microcycle Value
------------------------------------------------------------------------
function microcycle;
input: np      /* number of periodic variables */
       tp[i]   /* vector containing the periodicity of the variables */
output:  μCy     /* value of the microcycle */

begin
```

```
 1:    min = MAXINT;
 2:    for i = 1 to np do
 3:       if tp[i] < min then
 4:          min = tp[i]
 5:       end if
 6:    end for;
 7:    μCy = min + 1;
 8:    repeat
 9:       μCy = μCy - 1;
 10:      ctrl = TRUE;
 11:      for i = 1 to np do
 12:         if tp[i] mod μCy <> 0 then
 13:            ctrl = FALSE
 14:         end if
 15:      end for
 16:   until control = TRUE;
return μCy;
-----------------------------------------------------------------------
```

The macrocycle (*MCy*) is defined as:

$$MCy = N \times \mu Cy \tag{5}$$

where N is the number of microcycles that compose a macrocycle. Using the LCM rule, N can be evaluated as follows:

$$N = \min\{\Phi\} \wedge \Phi \in \aleph, \quad \text{with } \frac{\Phi}{Tp_i / \mu Cy} = \left\lfloor \frac{\Phi}{Tp_i / \mu Cy} \right\rfloor, \forall_i \tag{6}$$

An algorithm for the evaluation of the macrocycle value may be as follows:

```
-----------------------------------------------------------------------
- Evaluation of the Macrocycle Value
-----------------------------------------------------------------------
function macrocycle;
input: np      /* number of periodic variables */
       tp[i]   /* vector containing the periodicity of the variables */
       μCy     /* value of the microcycle */
output:  Mcy     /* value of the macrocycle */

begin
 1:    max = 0;
 2:    for i = 1 to np do
 3:       if tp[i] > max then
 4:          max = tp[i]
 5:       end if
 6:    end for;
 7:    N = max - 1;
 8:    ctrl = FALSE;
 9:    while ctrl = FALSE do
 10:         N = N + 1;
 11:      ctrl = TRUE;
 12:      for i = 1 to np do
 13:         if N mod (tp[i]/μCy) <> 0 then
 14:            ctrl = FALSE
 15:         end if
 16:      end for
```

```
 17:   end while;
 18:   MCy = N × μCy;
return MCy;
-------------------------------------------------------------------------
```

The BAT can be easily built according to a rate monotonic (RM) algorithm [12], enabling the specification of a feasibility test for the periodic traffic (sub-section 3.3).

Considering the WorldFIP characteristics, the BAT can be built as follows:

1. From variable with the shortest period until variable with the longest period

   1.1 If the load in each cycle plus the variable's length (buffer transfer length) is still shorter than the value of the microcycle, then schedule a scan for that variable in each one of the microcycles (of a macrocycle) multiple of the period of the variable. Update the value of the load in each concerned microcycle.

   1.2 If the load in some of the microcycles does not allow to schedule a scan for that variable, try to schedule it for the first of the subsequent microcycles up to the microcycle in which a new scan for that variable should be made. If this is not possible, the variable set is not scheduled.

For the example of table 1 (Cp_i = 0.0976ms, $\forall_i$) and considering the RM algorithm, the BAT is:

Table 2: BAT (using RM) for Example of Table 1

	Microcycle											
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
bat[*A*,*cycle*]	1	1	1	1	1	1	1	1	1	1	1	1
bat[*B*,*cycle*]	1	0	1	0	1	0	1	0	1	0	1	0
bat[*C*,*cycle*]	1	0	0	1	0	0	1	0	0	1	0	0
bat[*D*,*cycle*]	1	0	0	0	1	0	0	0	1	0	0	0
bat[*E*,*cycle*]	1	0	0	0	1	0	0	0	1	0	0	0
bat[*F*,*cycle*]	1	0	0	0	0	0	1	0	0	0	0	0

where *bat*[*i*, *j*] is a table of booleans with *i* ranging from 1 up to *np*, and *j* ranging from 1 up to *N* (number of microcycles in a macrocycle).

Below, we give a detailed description of an algorithm for building the BAT using the RM algorithm is presented. In the algorithm, the vector *load*[ ] is used to store the load in each microcycle as the traffic is scheduled. It also assumes that the array *Vp*[,] is ordered from the variable with the shortest period (*Vp*[1,]) to the variable with the longest period (*Vp*[*np*,]).

```
-------------------------------------------------------------------------
- Rate Monotonic for Building the BAT
-------------------------------------------------------------------------
function rm_bat;
input: np      /* number of periodic variables */
       Vp[i,j] /* array containing the periodicity of the variables */
               /* ORDERED by periodicities */
               /* i ranging from 1 to np */
               /* and the length of Cpi, j ranging from 1 to 2 */
       µCy     /* value of the microcycle */
       N       /* number of microcycles in the macrocycle */
output:
       bat[i,cycle] /* i ranging from 1 to np */
                    /* cycle ranging from 1 to N */
begin
 1:    for i = 1 to np do
 2:       cycle = 1;
 3:       repeat
 4:          if load[cycle] + Vp[i,2] <= µCy then
 5:             bat[i,cycle] = 1;
 6:             load[cycle] = load[cycle] + 1;
 7:             cycle = cycle + (Vp[i,1] div µCy)
 8:          else;
 9:             cycle1 = cycle1 + 1;
 10:            ctrl = FALSE;
 11:            repeat
 12:               if load[cycle1] + Vp[i,2] <= µCy then
 13:                  ctrl = TRUE
 14:               end if;
 15:            until (ctrl = TRUE) or (cycle1 >= (cycle + (Vp[i,1]
 16:                              div µCy)));
 17:            if cycle1 >= (cycle + (Vp[i,1] div µCy)) then
 18:               bat[i,cycle1] = 1;
 19:               load[cycle1] = load[cycle1] + 1;
 20:               cycle = cycle + (Vp[i,1] div µCy)
 21:            else
 22:               /* MARK Vpi NOT SCHEDULABLE with RM Algorithm */
 23:               cycle = cycle + (Vp[i,1] div µCy)
 24:            end if
 25:         end if
 26:      until cycle > N
 27:   end for
return bat;
-------------------------------------------------------------------------
```

Note that by using the RM algorithm some of the variables with longer periods can be scheduled in subsequent microcycles, thus inducing an increased communication jitter for those variables. For example, if the network data rate is 1Mbps instead of 2.5Mbps (Cp_i=(64+80)/1+2×20=184µs), the BAT would be as shown in table 3, since a microcycle is only able to schedule up to 5 periodic buffer transfers (figure 5).

Table 3: BAT (using RM) for Modified Example of Table 1

	Microcycle											
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
Bat[*A*,*cycle*]	1	1	1	1	1	1	1	1	1	1	1	1
Bat[*B*,*cycle*]	1	0	1	0	1	0	1	0	1	0	1	0
bat[*C*,*cycle*]	1	0	0	1	0	0	1	0	0	1	0	0
bat[*D*,*cycle*]	1	0	0	0	1	0	0	0	1	0	0	0
bat[*E*,*cycle*]	1	0	0	0	1	0	0	0	1	0	0	0
bat[*F*,*cycle*]	0	1	0	0	0	0	1	0	0	0	0	0

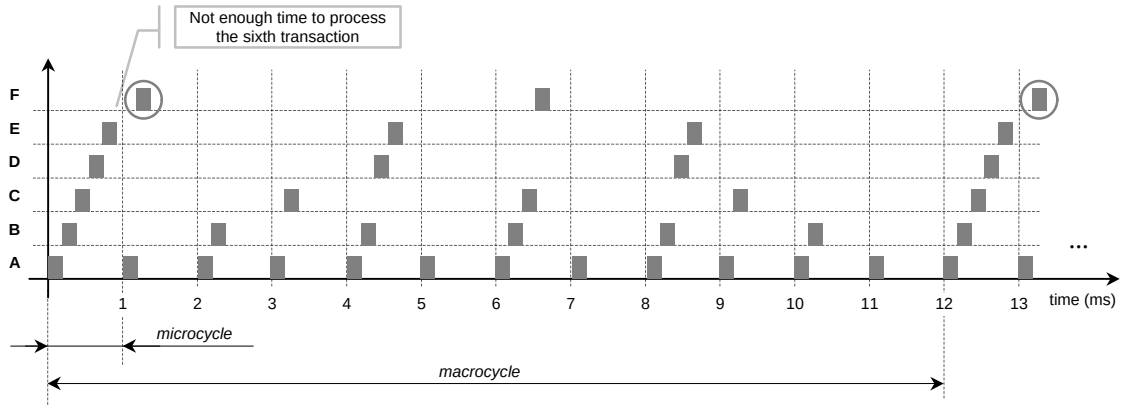

Figure 5

3.3. Feasibility Test for the Periodic Traffic

Although the *rm_bat* algorithm gives whether all traffic is schedulable or not (line 22), a simple pre-run-time schedulability test can be provided to check the schedulability of the periodic variable set.

The following analysis adapts to the WorldFIP case, worst-case response time analysis of tasks in a single processor environment presented in [13]. It constitutes an important step forward, since previous relevant work by Almeida *et al.* [14] was based on a modification of the RM utilisation-based test [12], thus embodying a non-negligible level of pessimism. Note that one of the disadvantages of utilisation-based tests, as compared to response time tests, is that they are sufficient but often not necessary tests. That is, if the variable set passes the test, the schedule will meet all deadlines; if it fails the test, the schedule may or may not fail at run time.

For the simplification of the analysis we consider that all transactions have a length of Cp, with $Cp=\max\{Cp_i\}$, $\forall_i$. In most of the cases this is a valid assumption, since the data field for these variables will concern process data which typically has a number of bytes ranging from 2 to 4.

The interference that a variable Vp_i suffers in NR_i microcycles is given by:

$$I_i = \sum_{j \in hp(i)} \left\lceil \frac{NR_i \times \mu Cy}{Tp_j} \right\rceil \quad \textbf{(7)}$$

where I_i corresponds to the number of "requests" for other variables with higher priority than Vp_i. Therefore, the number of requests to process during NR_i microcycles (including the request for Vp_i) is:

$$1 + I_i = 1 + \sum_{j \in hp(i)} \left\lceil \frac{NR_i \times \mu Cy}{Tp_j} \right\rceil \quad \textbf{(8)}$$

The maximum number of buffer exchanges that fit in a microcycle is given by:

$$\left\lfloor \frac{\mu Cy}{Cp} \right\rfloor \quad \textbf{(9)}$$

which is a constant. Therefore, the number of microcycles (NR_i) needed to process the request for variable Vp_i is given by:

$$NR_i = \min\{\Psi\} \wedge \Psi \in \aleph \wedge \Psi \le \frac{Tp_i}{\mu Cy}, \text{ with } 1 + \sum_{j \in hp(i)} \left\lceil \frac{\Psi \times \mu Cy}{Tp_j} \right\rceil \le \Psi \times \left\lfloor \frac{\mu Cy}{Cp} \right\rfloor \quad \textbf{(10)}$$

In (10), inequality $1 + I_i \le Y \times \lfloor \mu Cy/Cp \rfloor$ is tested iteratively, starting with $Y = 1$. If the solution (if any) gives $Y > Tp_i / \mu Cy$ then Vp_i is not schedulable.

Example:

Table 4: Example Set of Periodic Buffer Transfers (Cp = 0.21ms)

Identifier	**A**	**B**	**C**	**D**	**E**
Periodicity (ms)	1	1	1	1	3

Test for variable Vp_E:

$\Psi = 1$:

$$1 + \left\lceil \frac{1}{1} \right\rceil + \left\lceil \frac{1}{1} \right\rceil + \left\lceil \frac{1}{1} \right\rceil + \left\lceil \frac{1}{1} \right\rceil = 5 \le 1 \times 4, \quad \text{FALSE}$$

$\Psi = 2$:

$$1 + \left\lceil \frac{2}{1} \right\rceil + \left\lceil \frac{2}{1} \right\rceil + \left\lceil \frac{2}{1} \right\rceil + \left\lceil \frac{2}{1} \right\rceil = 9 \le 2 \times 4, \quad \text{FALSE}$$

$\Psi = 3$:

$$1 + \left\lceil \frac{3}{1} \right\rceil + \left\lceil \frac{3}{1} \right\rceil + \left\lceil \frac{3}{1} \right\rceil + \left\lceil \frac{3}{1} \right\rceil = 13 \le 3 \times 4, \quad \text{FALSE}$$

Which means that Vp_E is not schedulable, as Y must be smaller or equal than $Tp_E / \mu Cy = 3$

4. Conclusions

In this paper we provide a comprehensive study on how to set the WorldFIP bus arbitrator table. We provide tools and analysis for guaranteeing the real-time requirements of the data transfers.

The highest common factor (HCF) / least common multiple (LCM) methodology for setting the BAT is described in detail. Based on this methodology we show how the rate monotonic (RM) scheduling policy can be used to guarantee real-time behaviour of the WorldFIP network. We show also how the RM may induce non-negligible communication jitter in the scanning periodicities.

Important contribution is also made in the definition of a simple feasibility test (inequality (10)) for the periodic traffic scheduled according to the RM algorithm.

References

[1] Lenhart, G. A Fieldbus Approach to Local Control Networks. *Advances in Instrumentation and Control*, Vol. 48, No. 1, pp. 357-365, 1993.

[2] Cardoso, A. and E. Tovar. Industrial Communication Networks: Issues on Heterogeneity and Internetworking. *Proceedings of the 6th International*

Conference on Flexible Automation and Intelligent Manufacturing (FAIM'96), Atlanta, USA, pp. 139-148, Begell House Publishers, 1996.

[3] Cenelec. General Purpose Field Communication System. EN 50170, Vol. 1/3 (P-NET), Vol. 2/3 (PROFIBUS), Vol. 3/3 (FIP), Cenelec, 1996.

[4] Pnet. The P-NET Standard. International P-NET User Organisation ApS, 1994.

[5] Profibus. PROFIBUS Standard DIN 19245 part I and II. Translated from German, Profibus Nutzerorganisation e.V., 1992.

[6] Afnor. Normes FIP NF C46-601 to NF C46-607. Union Technique de l'Electricité, 1990.

[7] ISO 9506. Industrial Automation Systems - Manufacturing Message Specification. ISO, 1990.

[8] Afnor. Normes FIP NF C46-605 - Gestion de Réseau. Union Technique de l'Electricité, 1990.

[9] Kim, Y., Jeong, S. and W. Kown. A Pre-Run-Time Scheduling Method for Distributed Real-Time Systems in a FIP Environment. *Control Engineering Practice*, Vol. 6, pp. 103-109, Pergamon/Elsevier Science, 1998.

[10] Kumaran, S. and J.-D. Decotignie. Multicycle Operations in a Field Bus: Application Layer Implications. *Proceedings of IEEE Annual Conference of Industrial Electronics Society (IECON'89)*, pp. 531-536, 1989.

[11] Raja, P. and G. Noubir. Static and Dynamic Polling Mechanisms for Fieldbus Networks. *Operating Systems Review*, Vol. 27, No. 3, 1993.

[12] Liu, L. and J. Layland. Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment. *Journal of the ACM*, 20(1), pp. 46-61, 1973.

[13] Joseph, M. and P. Pandya. Finding Response Times in a Real-Time System. *The Computer Journal*, Vol. 29, No. 5, pp. 390-395, 1986.

[14] Almeida, L., Pasadas, R. and J. Fonseca. Using a Planning Scheduler to Improve the Flexibility of Real-Time Fieldbus Networks. *Control Engineering Practice*, 7, pp. 101-108, 1999.