

AB-Note-2005- 029(CO)

# Evaluation of a fast PLC module in prospect of the LHC beam interlock system

*Manuel Zaera Sanz*

## Abstract

The LHC Beam Interlock system requires a controller performing a simple matrix function to collect the different beam dump requests. To satisfy the expected safety level of the Interlock, the system should be robust and reliable. The PLC is a promising candidate to fulfil both aspects but too slow to meet the expected response time which is of the order of  $\mu$ seconds. Siemens has introduced a “so called” fast module (FM352-5 Boolean Processor) that provides independent and extremely fast control of a process within a larger control system using an onboard processor, a Field Programmable Gate Array (FPGA), to execute code in parallel which results in extremely fast scan times. It is interesting to investigate its features and to evaluate it as a possible candidate for the beam interlock system. This note publishes the results of this study. As well, this note could be useful for other applications requiring fast processing using a PLC.

## TABLE OF CONTENTS

<b>INTRODUCTION .....</b>	<b>3</b>
<b>PRODUCT OVERVIEW .....</b>	<b>4</b>
FUNCTIONS OF THE FM 352-5 BOOLEAN PROCESSORS .....	4
OPERATING CHARACTERISTICS .....	4
SYSTEM CONFIGURATIONS .....	4
PHYSICAL FEATURES OF THE MODULE .....	6
<b>PROGRAMMING THE FM 352-5 .....</b>	<b>10</b>
SOFTWARE AND HARDWARE REQUIREMENTS .....	10
CONFIGURING THE FM 352-5 .....	10
PROGRAMMING AND OPERATING THE FM 352-5 .....	12
<b>BEAM INTERLOCK SYSTEM: CONTROL AND RESPONSE TIME ANALYSIS .....</b>	<b>15</b>
CONTROL PROGRAM .....	17
LABORATORY PLC AND MEASUREMENT EQUIPMENT .....	18
RESPONSE TIME ANALYSIS .....	19
SINGLE MODULE CONFIGURATION .....	19
CHAINED MODULES CONFIGURATION .....	24
MANAGING MASKS .....	26
ARMING THE FM 352-5 .....	26
ANALYSIS OF EXTERNAL CLOCK SIGNAL GENERATION USING THE FM352-5 ...	27
RELIABILITY ISSUES OF THE FM352-5 .....	28
<b>BEAM INTERLOCK SYSTEM: MONITORING AND PROGRAMMING.....</b>	<b>30</b>
PROGRAMMING ELEMENTS OFFERED BY THE MODULE .....	30
ANALYSING COUNTERS WITH AN EXTERNAL CLOCK GENERATOR FOR TIME INTERVAL MEASUREMENT .....	31
TIME INTERVAL MEASUREMENT .....	32
DATA STRUCTURES .....	32
PROGRAMMING PRIMITIVES .....	32
USING FREQUENCY DIVISORS .....	35
DOUBLE BUFFERING TECHNIQUE AND ITS FM352-5 IMPLEMENTATION .....	37
ABSOLUTE TIME STAMPING ARCHITECTURES .....	39
PROGRAM IDENTIFICATION .....	44
BOOTING THE FM352-5 WITH A PROGRAMMED MMC .....	45
<b>FURTHER WORK.....</b>	<b>45</b>
<b>ACKNOWLEDGEMENTS .....</b>	<b>46</b>
<b>REFERENCES.....</b>	<b>46</b>

## INTRODUCTION

The LHC beam interlock system [1] requires a controller performing a simplex matrix function to collect the different beam dump requests. These requests come from different equipments such as beam loss monitors, access, experiments, vacuum, RF, beam dump, etc.

To satisfy the expected safety level for the interlocks, the system should be robust and reliable. One of the most reliable and robust systems are those based on PLCs (*Programmable Logic Controllers*) [1][2] being them possible candidates for the implementation of the beam interlock system.

One of the main problems is the  $\mu seconds$  response time requirement. This requirement is imposed by the LHC beam turn which duration is  $88\mu s$ . If any equipment detects a failure, we must be sure that the beam is dumped as fast as possible. The control operation must be done with hard real-time constraints, that is, the process must be controlled within a deadline. This deadline is two turns to the LHC, otherwise the system could be damaged.

The control implementation program requires execution times of the order of  $\mu seconds$ . A standard PLC CPU cannot fulfill this requirement, and we need a fast module for performing this control operation. *Siemens* supplies the module known as FM352-5 boolean processor. It provides independent and extremely fast control of a process within a larger control system using an on-board processor, a FPGA (*Field Programmable Gate Array*), to execute code in parallel which results in extremely fast scan times.

Another problem to solve is related with postmortem analysis. It is important to know which equipment has requested the beam dump, at which time, and time differences between several dump requests. A monitoring application is required, which main function is the storage of the equipment that issues the request and the associated time stamp.

The purpose of this note is the evaluation of the FM352-5 Boolean processor in order to verify if it is able to meet the requirements imposed by the LHC beam interlock. For the control application we have to measure the response time provided by the module, and for the monitoring application we have to solve several problems related to time stamping such as the storage, access from external equipments, standard format and get back the records. Besides, the present study could be useful for other applications requiring fast process using PLCs.

The document is organized as follows. The first two sections detail the main features of the Boolean processor and its programming, the following section consists of a detail study of the response time offered by the module related to the control requirements, and the last section is focused on the monitoring application and its programming. The note ends with the further work, acknowledgements and references.

## **PRODUCT OVERVIEW**

In this section we give an overview of the FM 352-5 focusing on the features that could provide interesting information about its functionality and the feasibility using it for the beam interlock system.

### ***FUNCTIONS OF THE FM 352-5 BOOLEAN PROCESSORS***

The FM 352-5 is a high-speed Boolean processor that provides independent and extremely fast control of a process within a larger control system.

The FM 352-5 module uses an onboard processor, a Field Programmable Gate Array (FPGA), to execute code in parallel rather than sequentially as standard programmable controllers do. This type of execution results in extremely fast scan times.

The module controls a number of built-in input and outputs (up to 15 inputs and 8 outputs). In addition to the normal I/O points, the module can support one of three encoder types (incremental differential, 24 V single-ended, and SSI *Synchronous Serial Interface* absolute encoders). If no encoder interfaces are used, the differential pins are available to provide three discrete differential (5V) inputs (numbers 12, 13, and 14).

### ***OPERATING CHARACTERISTICS***

The FM 352-5 module executes its program independently of the PLC CPU. The inputs and outputs of the process controlled by the module are local and cannot be directly accessed by the PLC CPU. However, the user program of the PLC CPU transfers control commands and configuration parameters to the FM 352-5 module over the I/O bus and evaluates the status information returned by the module.

The Boolean processor has the following operating characteristics:

- Recording and control of fast processes (for example, high-speed inspection & rejection systems, or control of high speed machines in the packaging, food & beverage, tobacco, and personal care product industries).
- Data exchange with the PLC CPU user program (when used in a coprocessor configuration). Point 1.3 reviews the possible configurations of the Boolean processor).

### ***SYSTEM CONFIGURATIONS***

The FM 352-5 module can be configured to operate in the following ways:

- Coprocessor configuration within an S7 programmable controller system. In this configuration, the FM 352-5 exchanges input/output data, and status and control information with the PLC CPU as it is shown in Figure 2.1 and 2.2.
- The FM 352-5 module can also operate as a stand-alone controller independently of any PLC system as it is shown in Figure 2.3.

Evaluation of a fast PLC module in prospect of the LHC beam interlock system

- In a distributed configuration, the FM 352-5 module functions as a module of an ET200M normal PROFIBUS-DP slave to an S7 or non-S7 master as it is shown in Figure 2.4.

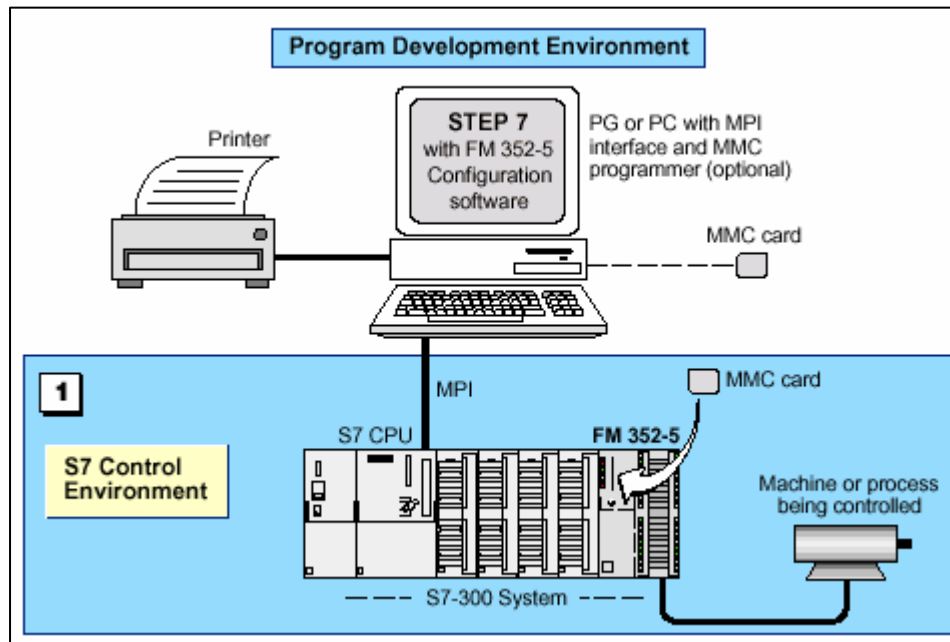


Figure 2.1: FM 352-5 operation in coprocessor configuration: Program development environment

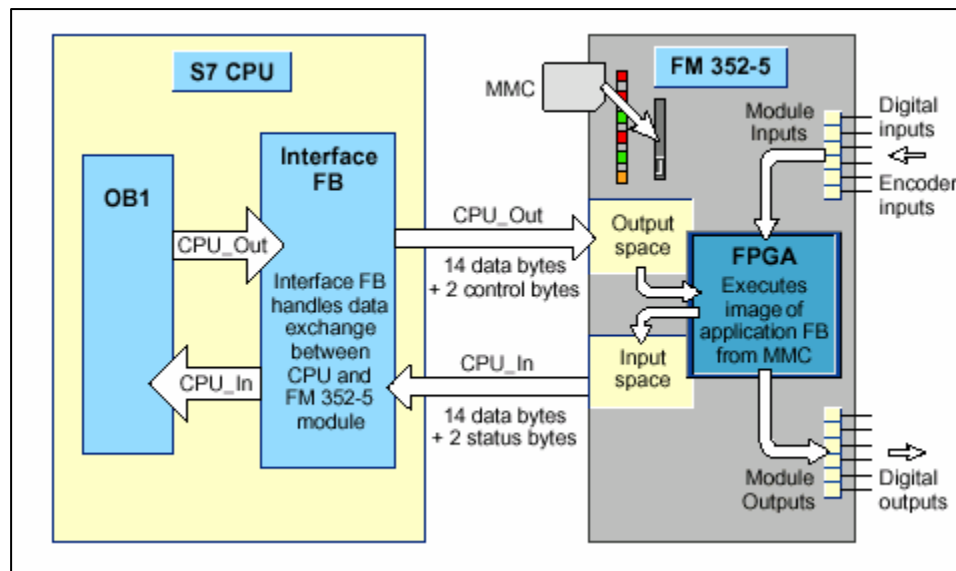


Figure 2.2: FM 352-5 Operation in Coprocessor Configuration: Programming interface.

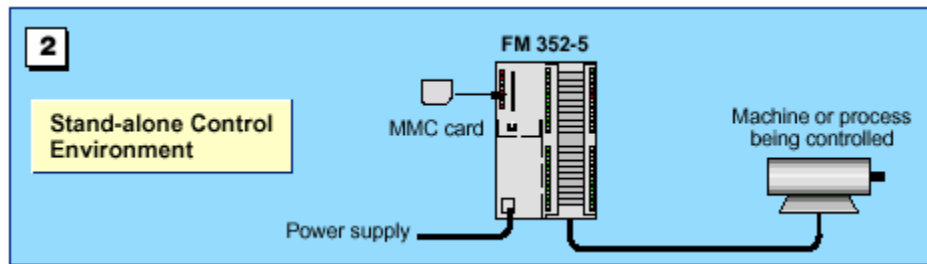


Figure 2.3: FM 352-5 Operation as stand-alone controller.

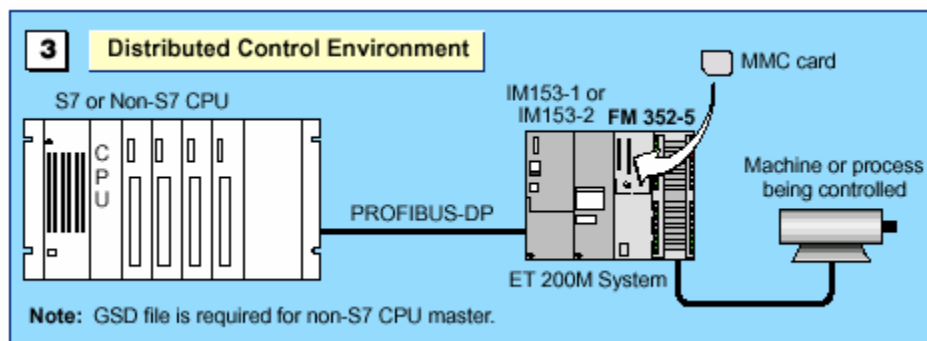


Figure 2.4: FM 352-5 Operation as PROFIBUS-DP slave.

## PHYSICAL FEATURES OF THE MODULE

1. The Status Indicators: The Status leds on the faceplate of the FM 352-5 module indicate the following conditions:

LED Label	LED	Color	Description
SF		Red	Indicates a fault condition in the module.
MCF		Red	Indicates a fault condition in the MMC of the module.
DC5V		Green	Indicates the power status of the module.
I0F		Red	Indicates an I/O fault condition: output overload, missing 2L or 3L, broken wire, SSI fault.
RUN		Green	Indicates the module is in RUN mode.
STOP		Yellow	Indicates the module is in STOP mode.
I0 to I11		Green	Indicates the On status of each input point.
Q0 to Q7		Green	Indicates the On status of each output point.
5VF		Red	Indicates an overload in the 5 V power supply output.
24VF		Red	Indicates an overload in the 24 V power supply output.

Table 2.1: Status LED definitions

Usually when the module works in the co-processor configuration and the MMC is not inserted (or it is inserted without a program), the SF and MCF leds are on. This is not a fatal error due to a missing memory card is a (low priority) fault because with the next power off/on the program in the FM will be lost. In any case it is important to check the status word (Table 2.2) of the standard FB in order to know the origin of the error pointed by the LEDS

Bits	Command to Module	Bits	Response from Module
<b>Operating Mode</b>		<b>Operating Status</b>	
0000	Continue current normal mode	0001	Normal mode — STOP
0001	Normal mode — STOP	0010	Normal mode — RUN
0010	Normal mode — RUN	0101	Debug mode — STOP (outputs off)
0101	Debug mode — STOP	0110	Debug mode — RUN
0110	Debug mode — RUN	1010	Single scan mode
1010	Single scan mode — SCAN once*		
1000	Single scan mode — no change (idle)		<b>MMC Status</b>
		000	MMC good
		001	No MMC present
		010	Bad or invalid MMC
		011	MMC program missing
		100	MMC program corrupted
		111	MMC and Data Record 0/128 do not match (applies to S7 masters only)

\* If the Single Scan bit is set to 1, the module executes one scan when the RUN bit transitions from 0 to 1.

Table 2.2: Bit definitions of the control and status bytes.

The status leds have the behavior depicted on Table 2.3 according to each of the listed operations being executed.











Active LEDs	LED	Behavior	Operation
All LEDs	   	On for 1 second	LED test at power-up.
RUN STOP	 	Fast blink (2 Hz) On	When the module is receiving a download from the MMC or the PC.
RUN STOP	 	Slow blink (0.5 Hz) Off	When module is in Debug/RUN mode.
RUN STOP	 	Slow blink (0.5 Hz) On	When module is in Debug/STOP mode.

Table 2.3: Behaviour of status leds according to operation

## 2. Other Physical Features:

- Three-position switch to set the operating mode of the module. The reset position (MRES) is spring-loaded with no detent.

## Evaluation of a fast PLC module in prospect of the LHC beam interlock system

- Slot for the MMC (Micro Memory Card), which stores the program in non-volatile memory. The FM352-5 records this MMC in the download operation. (if inserted).
  - Removable terminal connector for wiring inputs and outputs. It is a 40 pin Terminal connector for input and output signals to the module which order number is 6ES7 392-1AM00-0AA0.
3. Micro Memory Card: The MMC stores the program files in non-volatile memory, and installs in the slot on the front of the FM 352-5 module. The possible sizes of an MMC are: 128 Kbytes, 512 Kbytes (which order number is 6ES7 953-8LJ00-0AA0) or 2 Mbytes of memory.

The module could work as a co-processor without MMC, however in the stand-alone configuration it is needed for operation.

If the programmed MMC is installed in the module at power-up, the FM352-5 copies its program from the MMC to the FPGA, sets Normal mode and enters operating state STOP. With no programmed MMC installed, the FM 352-5 copies its internal program to the FPGA, sets normal mode and enters operating state STOP. If configured to operate in a co-processor environment, subsequent mode and operating state transitions are determined by the appropriated interface FB in conjunction with the RUN/STOP switch located on the FM 352-5 front panel.

Currently, the FM352-5 reference 352-5AH00-0AE0 HW v1 Firmware v2.0.0 has a bug that prevents it working properly with the MMC of 512kbytes.

4. Wiring Diagram: A simplified wiring diagram is provided on the inside of the terminal connector door as shown in Figure 2.5

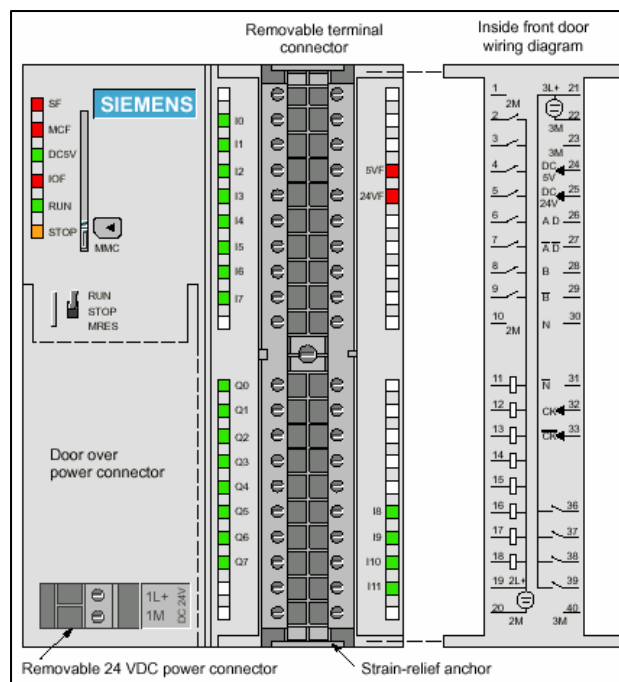


Figure 2.5: Front terminal connector of the FM 352-5



## Evaluation of a fast PLC module in prospect of the LHC beam interlock system

The inputs, outputs, encoder signals, and input/output power supply wiring are all connected to the 40 pin terminal connector, which installs under the hinged door. On the bottom left side of the module, under a hinged cover door, are the 1L+ and 1M terminal connections for the 24 VDC power supply wiring for the module logic circuitry. This connection together with 2L+/2M, are the minimum wiring connections required to start up the FM 352-5 module. If we want to use the inputs I12, I13 and I14, we must wire the 3L+/3M terminal connections to the 24 VDC.

Working without encoders the module provides:

- 15 digital inputs: From I0 to I11 are 24V digital inputs, and from I12 to I14 are differential 5V digital inputs.
- 8 sinking outputs. From Q0 to Q7.

which pin assignments are depicted on Tables 2.4 and 2.5.

Pin #	I/O	Name	Function	LED
1		2M	Ground for section 2 – input/output circuitry	—
2	Input	I 0	Input	Green
3	Input	I 1	Input	Green
4	Input	I 2	Input	Green
5	Input	I 3	Input	Green
6	Input	I 4	Input	Green
7	Input	I 5	Input	Green
8	Input	I 6	Input	Green
9	Input	I 7	Input	Green
10		2M	Ground for section 2 – input/output circuitry	—
11	Output	Q 0	Sinking output	Green
12	Output	Q 1	Sinking output	Green
13	Output	Q 2	Sinking output	Green
14	Output	Q 3	Sinking output	Green
15	Output	Q 4	Sinking output	Green
16	Output	Q 5	Sinking output	Green
17	Output	Q 6	Sinking output	Green
18	Output	Q 7	Sinking output	Green
19		2L+	Power for section 2 – input/output circuitry	—
20		2M	Ground for section 2 – input/output circuitry	—

Table 2.4: Terminal connector assignments. Pins 1 to 20

Pin #	I/O	Name	Encoder Function				LED
			5 V Encoder	SSI Master	SSI Listen	24 V Encoder	
21		3L+	Power for section 3 – encoder circuitry				—
22		3M	Ground for section 3 – encoder circuitry				
23		3M	Ground for section 3 – encoder circuitry				
24	Output	5V Out	5.2 V encoder supply				Red
25	Output	24V Out	24 V encoder supply				Red
26	Input	Encoder	Phase A	Master SSI D (data)	Listen SSI D (data)	I 12+	
27	Input	Encoder	Phase A (inverse)	SSI D (data inverse)	SSI D (data inverse)	I 12–	
28	Input	Encoder	Phase B	I 13+	SSI CK (shift clock)	I 13+	
29	Input	Encoder	Phase B (inverse)	I 13–	SSI CK (shift clock inverse)	I 13–	
30	Input	Encoder	Marker N	I 14+	I 14+	I 14+	
31	Input	Encoder	Marker N (inverse)	I 14–	I 14–	I 14–	
32	Output	Encoder	—	SSI CK (shift clock)	—	—	
33	Output	Encoder	—	SSI CK (shift clock inverse)	—	—	
34	—	—	—	—	—	—	
35	—	—	—	—	—	—	
36	Input	I 8	I 8	I 8	I 8	I 8	Green
37	Input	I 9	I 9	I 9	I 9	Phase A	Green
38	Input	I 10	I 10	I 10	I 10	Phase B	Green
39	Input	I 11	I 11	I 11	I 11	Marker N	Green
40		3M	Ground for section 3 – encoder circuitry				—

Table 2.5: Terminal connector assignments. Pins 21 to 40

## **PROGRAMMING THE FM 352-5**

In this topic we study the requirements for programming the FM 352-5 within the STEP 7 environment. We study also how to configure the module and how to program it using the FM 352-5 library functions.

### ***SOFTWARE AND HARDWARE REQUIREMENTS***

The useful programming environment software that has to be pre-installed in the computer used for programming the FM 352-5 consists of:

- Step 7 v5.1 & SP 3
- NCM S7 PROFIBUS v5.1 & SP3
- NCM S7 Industrial Ethernet v5.1 & SP 3
- S7-SCL v5.1 & SP 2
- S7-GRAPH v5.1
- S7-PLCSIM v5.0 & SP1
- UCL
- SIMATIC NET v6.0

Afterwards, we have to install the FM 352-5 software that contains the next components:

- FM 352-5 Hardware Configuration software (including help files and compiler)
- FM 352-5 library of function blocks (FBs) and associated help files
- User Manual in PDF format
- GSD file (contains module parameter data for non-S7 masters)
- Example programs
- S7-PLCSIM.

The FM 352-5 Hardware Configuration software and the associated files are intended to work with SIMATIC STEP 7. If your computer meets the hardware requirements to support STEP 7, then your computer will also support the installation of the FM 352-5 Hardware Configuration software.

The FM 352-5 Hardware Configuration software operates with Windows 98, Windows NT and Windows 2k.

### ***CONFIGURING THE FM 352-5***

The basic flow of tasks and tools required for generating and downloading an application program for the FM 352-5 are (see Figure 3.1):

1. Create a HW configuration in the STEP7 HW Config Application.
2. Create the Application FB for the FM 352-5 in the STEP 7 LAD/FBD editor
3. Assign parameters to the FM 352-5 module in the properties dialog
4. Compile the application FB and HW configuration in the FM 352-5 properties dialog to generate an SDB for the FM 352-5 module.
5. Save and Compile the HW configuration in STEP 7 to generate a system data block for the CPU
6. From STEP 7, download the program blocks and the system data to the CPU
7. From the FM 352-5 properties dialog Programming tab, download the SDB, which contains the application FB and the module parameters, to the FM 352-5 module.

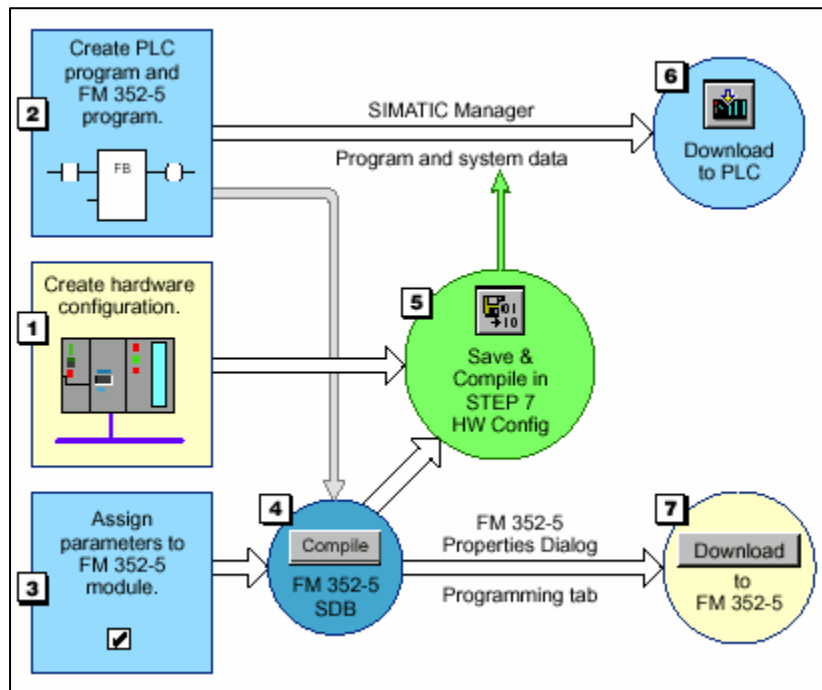


Figure 3.1: Tasks at a glance

The module FM 352-5 is configured by assigning certain properties and parameters through the Properties Dialog window which contains the next four tabs:

- *General*: where you could enter comments or descriptive information.
- *Addresses*: displays and allows you to change the system selected address assignments for the inputs and outputs. These addresses depend on the PLC HW which you are working with.
- *Setting Module Parameters*: provides a hierarchical view of the different functions and diagnostics of the FM 352-5. These parameters are grouped in 10 categories: Basic Parameters (Reaction to CPU STOP, and Interrupt generation and selection), Module Diagnostics Enable (Missing auxiliary supply voltage, Missing input/output supply voltage, MMC diagnostic, etc.), Output Diagnostics Enable (Output Overloads from Q0..Q7), Process Interrupt Enable (8 Process Interrupts), Input Filter Time Constants (15 Inputs Delays), Program Properties (Stand-Alone Operation and Consistency Check), Encoders Properties and Advanced Parameters (HW support for the Module Diagnostics, Output Diagnostics and Process Interrupts).  
Currently, the FM352-5 reference 352-5AH00-0AE0 HW v1 Firmware v2.0.0 has a bug that prevents it working properly with the *Consistency check* option that has to be disabled.
- *Programming*: This tab allows to program the FM 352-5 module specifying the FB number of the application. From this tab, you could: Edit your application, perform a Syntax check, Compile it, Download it to the FM 352-5, and obtain Module Information (Operating Mode, HW and Firmware version and Diagnostic Interrupts state).

## **PROGRAMMING AND OPERATING THE FM 352-5**

To create a program for the FM 352-5 we have to follow the next set tasks:

- Create the Application FB/DB
- Set up the Interface FB/DB set in OB1
- Debug the Application program
- Download the program to the FM 352-5 module
- Use STEP 7 to copy the program to the MMC with the MMC programming device.

The FM 352-5 Library contains two Interface FBs that allow the S7 CPU user program (OB1, for example) to control the mode and operating states of the FM 352-5 module. You need to insert a call in OB1 to the appropriate Interface FB that handles the exchange of data between the CPU and the FM 352-5 module.

The FM 352-5 could work in two modes: Debug mode and Normal mode. The transition from Normal to Debug mode is initiated by the CPU user program calling the Debug Interface FB (FB30 in the FM 352-5 Library). As a result of this mode transition command, the FM 352-5 replaces the program in the FPGA with its internal debug program.

To debug an application FB using the S7 CPU with the FM 352-5 module in debug mode, download to the CPU :

- The blocks in the regular CPU program
- The application FB, the one containing the FM 352-5 program, with its up-to-date instance DB.

FM interface debug FB (Figure 3.2) and its instance DB (FB30/DB30)

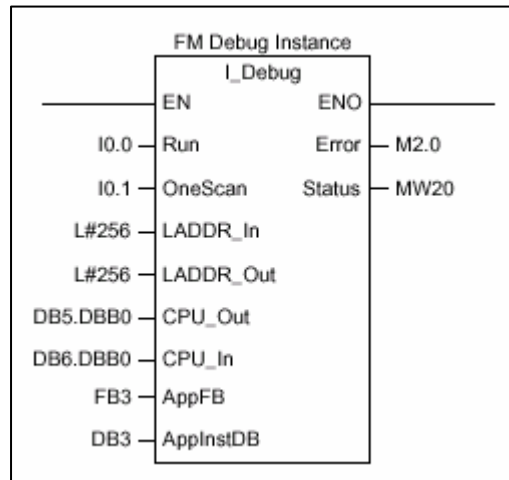


Figure 3.2: Interface FB for debug mode execution

In Debug mode, all program execution is performed by the S7 CPU, which allows the use of various program monitoring and debugging capabilities of STEP 7 to test the application program. The FM 352-5 module operates in a pass-through mode, making its inputs and outputs directly available to the S7 CPU.

## Evaluation of a fast PLC module in prospect of the LHC beam interlock system

The transition from debug to normal can be initiated by clicking the download button on the FM 352-5 configuration software programming tab. When the download to the FM 352-5 begins, the module enters operating state STOP and copies the downloaded file to the FPGA.

The MMC is not changed by the download. The FM 352-5 module remains in normal mode when the download completes and maintains operating state STOP until the CPU user program calls the normal interface FB (FB31 Figure 3.3) with a 1 at the Run input and the RUN/STOP switch in the RUN position. With this call the FM 352-5 module starts to execute the program that was downloaded to the FPGA.

The interchange data structures between the CPU and the FM 352-5 module have a length of 14 bytes, and are known as CPU\_OUT (CPU S7 =>FM 352-5) and CPU\_IN (CPU S7 <= FM 352-5). In Figures 3.2 and 3.3 these structures are called by the pointers DB5.DBB0 (for CPU\_OUT) and DB6.DBB0 (for CPU\_IN).

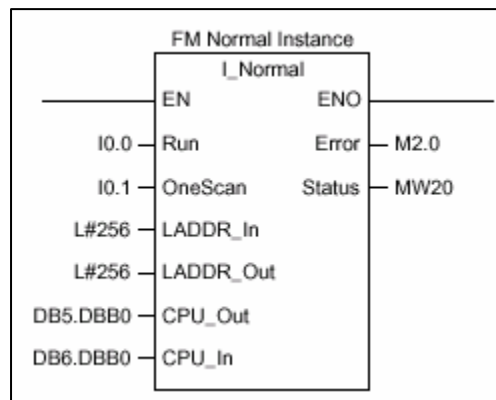


Figure 3.3: Interface FB for normal mode execution

The valid instructions from STEP 7 for the FM 352-5 are:

- Normally open or closed inputs
- NOT
- Output coil
- Midline output *Connector*
- SR and RS Flip-Flop
- Positive and negative RLO (*Result of Logic Operation*) edge detection
- Negative and positive Edge detection
- Comparators for INT and DINT
- Converters: from INT to DINT and MOVE

Additionally, the module has its own library of FB that could be called from the program:

- Binary scaler (FB 112)
- Timers of 16 and 32 bits (FB113..FB118)
- Clock pulse generator (FB119)
- Counters of 16 and 32 bits (FB120..FB123)
- Shift registers from 1 to 8 bits and lengths from 4096 (1 bit shift register) to 512 (8 bits shift register) (FB 124..FB127)
- Absolute value operation for 16 and 32 operands (FB104..FB105)

## Evaluation of a fast PLC module in prospect of the LHC beam interlock system

- Data selector for 16 bits and 32 bits operands (FB106..FB107)
- Subtract operation for 16 bits and 32 bits operands (FB108..FB109)
- Multiply operation for 16 bits and 32 bits operands (FB100..FB101)
- Divide operation for 16 bits and 32 bits operands (FB102..FB103)

The FM 352-5 module uses an onboard processor (the FPGA) to execute code in parallel rather than sequentially as standard programmable controllers do. This type of execution results in extremely fast and stable scan times.

The Multiphase clocking is a technique designed into the FM 352-5 translator software to manage the correct time sequencing of retentive elements relative to *connectors* (similar to marks) in the different networks of the application program. Twelve clock phases are available, eleven to clock elements with storage (timers, counters, flip-flops, edge detectors, shift registers and binary scalars), and the twelfth to clock the outputs. The module's 12-phase clock uses the *connectors* to synchronize the execution of previous or subsequent elements in the instruction networks.

The use of 12-phase clocking means you can connect up to 11 storage elements in series without worrying about extending the scan time. Inserting too many elements in series, generates an error message which helps to take the necessary action to meet the phase clock rules.

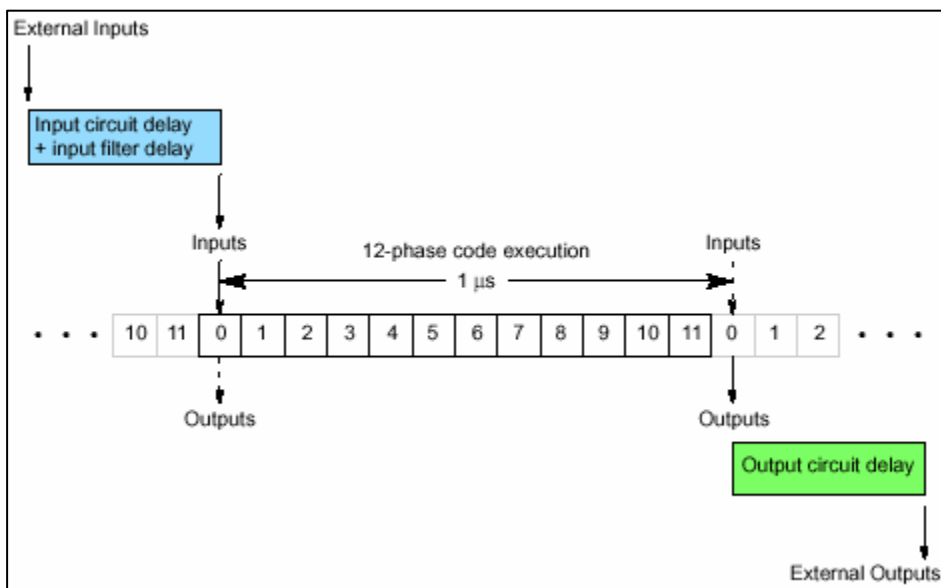


Figure 3.4: Multiphase clocking and I/O timeline

Figure 3.4 shows a graphic representation of how inputs and outputs are handled by the multi-phase clock execution of the FM 352-5 module. The total response time is calculated by adding the input delays, scan time, and output delays.

## BEAM INTERLOCK SYSTEM: CONTROL AND RESPONSE TIME ANALYSIS

In this section we discuss the beam interlock system requirement that consists of the status evaluation of the equipments such as beam loss monitors, access, experiments, vacuum, RF, beam dump, collimators, warm magnets, etc. If a failure is detected, a beam dump request is generated.

Figure 4.1 depicts the layout of the beam interlock system, where we have 8 IP (Insertion Points). Right and left of each IP there is a BIC (Beam Interlock Controller) having a total of 16 BICs. The BICs are interconnected by two loops (2 fast links) known as beam permit loops. When the loops are broken (open), the beams are extracted by the beam dump system. The two loops distinguish between beam one and beam two allowing independent control of each beam.

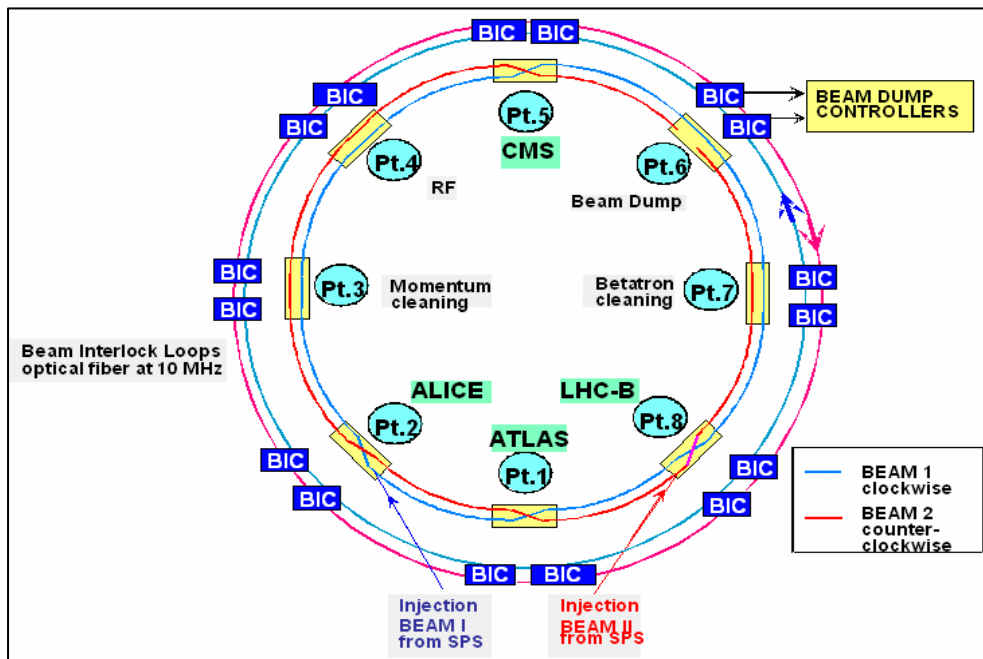


Figure 4.1: Layout of the LHC beam interlock system.

The architecture of the BIC is depicted in Figure 4.2. The objective is to combine the messages from different systems (vacuum, warm magnets, experiments, RF, etc.) to one output: BEAM PERMIT yes or not.

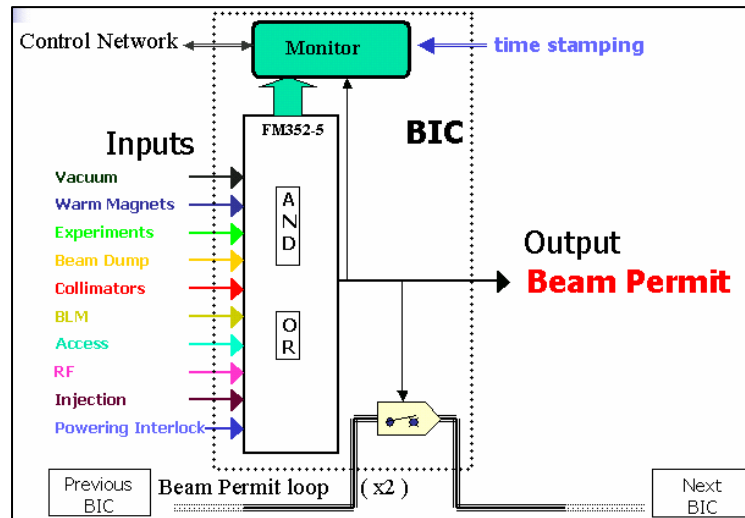


Figure 4.2: Architecture of the beam interlock controller.

Inside the Beam Interlock Controller we distinguish:

- The FM352-5 module which combines the different inputs for the output generation. The logic function it performs could be an AND or an OR operation. The choice depends on the logic levels used by the inputs for its activation or deactivation. (if they are 1 for activation the logic function is an OR or if they are 0 the logic function is an AND).
- The next element is the switch: when the Beam permit response is YES, the switch is closed. When the Beam permit response is NOT, the switch opens breaking the associated Beam permit Loop. Actually the switch is replicated having two switches one per Beam. The Beam permit Loop comes from the previous BIC and feeds the next BIC.
- The last element is the monitor system which function is to provide to external equipments connected through the Control network the accurate state of the LHC area controlled by that BIC. For this purpose it is needed a timing system that allows time stamping of different events detected by the FM352-5 module, and the output generated by the FM352-5 for Beam dump requests.

Our study is focused on the evaluation of the FM352-5 module as a solution for the implementation of the heart of the BIC.

In this section we address only the control operation of the Beam Interlock system and the response time analysis. In the next section we will address the problem related with Time Stamping.

Firstly we present the control program, next the laboratory PLC and the measurement equipment, the response time analysis for single and chained boolean processors, managing masks, arming the module, the analysis of external clock signal generation, and finally the reliability issues of the FM352-5 boolean processor.



## CONTROL PROGRAM

Because the requirement is a simple matrix, the control program is very simple and consists of an AND and an OR gate with a set of inputs each one (up to 15 inputs per FM 352-5 module).

The control program is shown in Figures 4.3 and 4.4 where we have a set of contacts in serial for the AND gate or in parallel for the OR gate implementation.

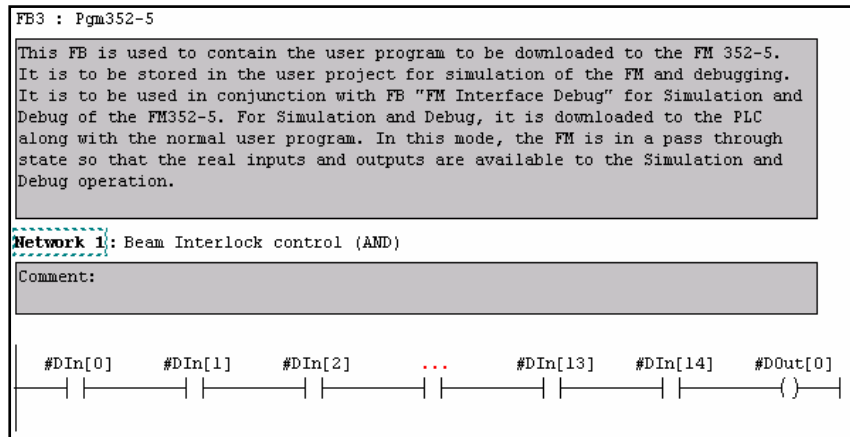


Figure 4.3: Beam Interlock Control (AND)

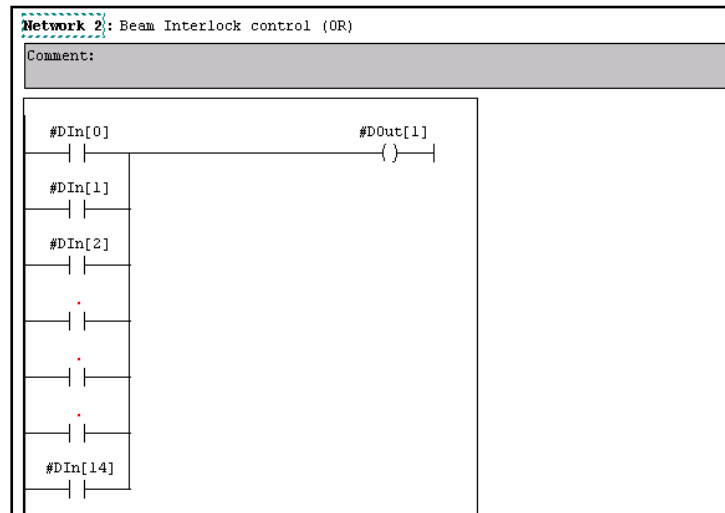


Figure 4.4: Beam Interlock Control (OR)

## **LABORATORY PLC AND MEASUREMENT EQUIPMENT**

The Laboratory PLC we have used to evaluate the FM 352-5 is composed by the next elements:

- SIEMENS PLC S7-300: Power supply PS307 5A, CPU 315, CP343-1 ethernet module, SM374 I/O module, FM352-5 module.
- TEKTRONIK TDS 2014 four channel digital oscilloscope & WaveStar v2.6 software for oscilloscopes.
- HP 3310A function generator
- LAMBDA LPT7202-FM regulated power supply.

It is important to describe in detail the HW/SW configuration of the SIEMENS PLC S7-300 related to the CP343-1 Ethernet module and the SM374 I/O module. The former allows to communicate with the PLC through the Ethernet Network, and the latter allows to simulate external input/output generation.

Under certain circumstances it is possible to start up automation systems exclusively via IE (*Industrial Ethernet*). Then, node initiation via MPI (*MultiPoint Interface*) is no longer necessary. The following CPs with IE interface can be used:

- CP343-1 with EX11 release 02 and higher, FW 2.0.0, at least STEP 7 v5.1+SP2
- CP343-1 IT with GX11 release 01 and higher, FW 2.0.7, at least STEP 7 v51+SP3
- CP443-1 with 6GK7 443-1EX11-0XE0 release 3 and higher, FW v2.0
- CP443-1 IT with 6GK7 443-1GX11-0XE0 release 2 and higher, FW v2.0

The configuration for direct communication between the PC and the PLC through the IE follows the next steps:

1. We can not initially configure routers (same physical network)
2. We must Set the PC/PG Interface: ISO Ind. Ethernet to our Ethernet card, and the Access Point of the Application: S7ONLINE (STEP7) to ISO Ind. Ethernet to our Ethernet card.
3. Creation and Configuration of our STEP7 project with our HW (see Figure 4.5).
4. Definition and Configuration of our automation system network using the NETPRO program (see Figure 4.6).

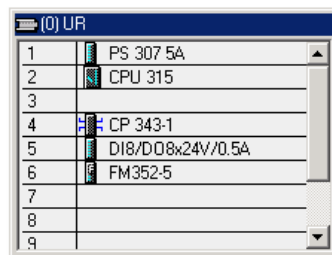


Figure 4.5: HW configuration

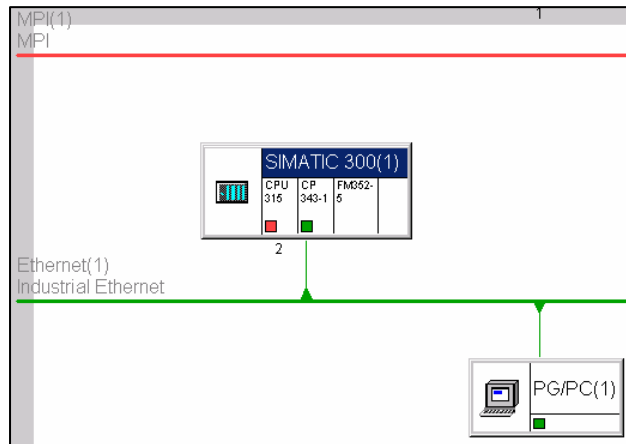


Figure 4.6: Network configuration

The SM374 I/O module is a simulation module, that is, this module does not appear at the hardware catalog because it replaces another module. We have to declare the module simulated by the SM374. Examples of simulated modules are:

- 6ES7 321-1BH00-0AA0: DI16xDC24v
- 6ES7 323-1BH00-0AA0: DI8/DO8x24v/0.5A

## RESPONSE TIME ANALYSIS

In this topic we evaluate the response time of the boolean processor in two different configurations. The first is a single module configuration in which we have only one module, and the second is a chained configuration with two chained modules

### SINGLE MODULE CONFIGURATION

To generate the stimulus to our matrix, we have built an input circuit depicted in Figure 4.7 that allows the generation of input signals just using simple switches. The state of these signals could be checked using the Oscilloscope connecting the probes as depicted in the figure.

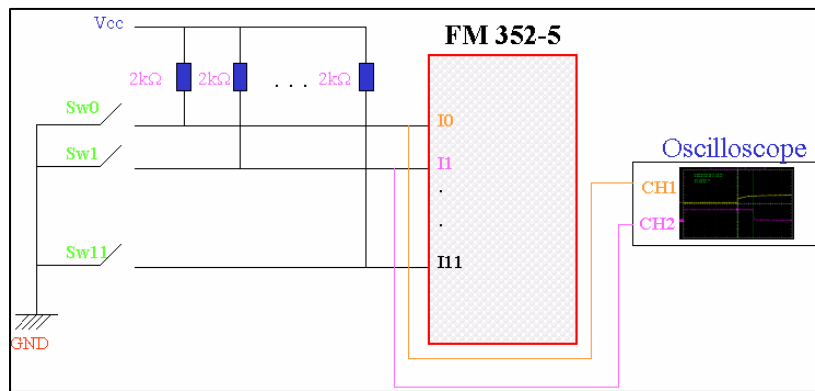


Figure 4.7: Inputs generation

In the case of the Outputs, we have to build a Pull-up circuit in order to read their activation and deactivation using the oscilloscope as is depicted in Figure 4.8.

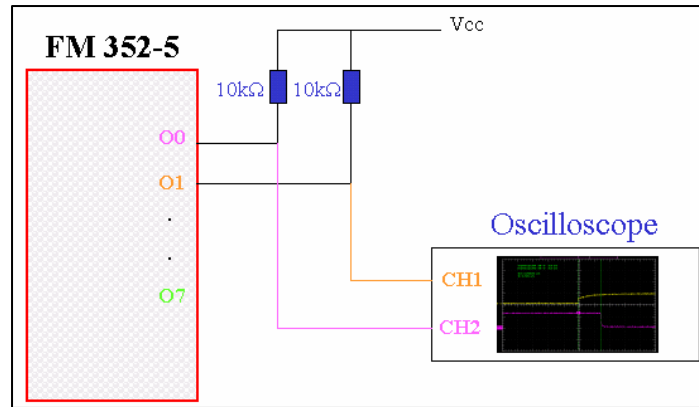


Figure 4.8: Pull-up circuit for outputs checking.

The conclusions we have obtained relative to the Response Time are the next ones:

- The number of inputs to an AND or to an OR gate does not affect the response time. This conclusion has been obtained analyzing the response time of an AND gate of two inputs and 12 inputs, and an OR gate of two inputs and twelve inputs (see Figures 4.9, and 4.10). We have obtained a response time of approximately  $3.6\mu\text{s}$  in the case of the AND gate, and  $3\mu\text{s}$  in the case of the OR gate.

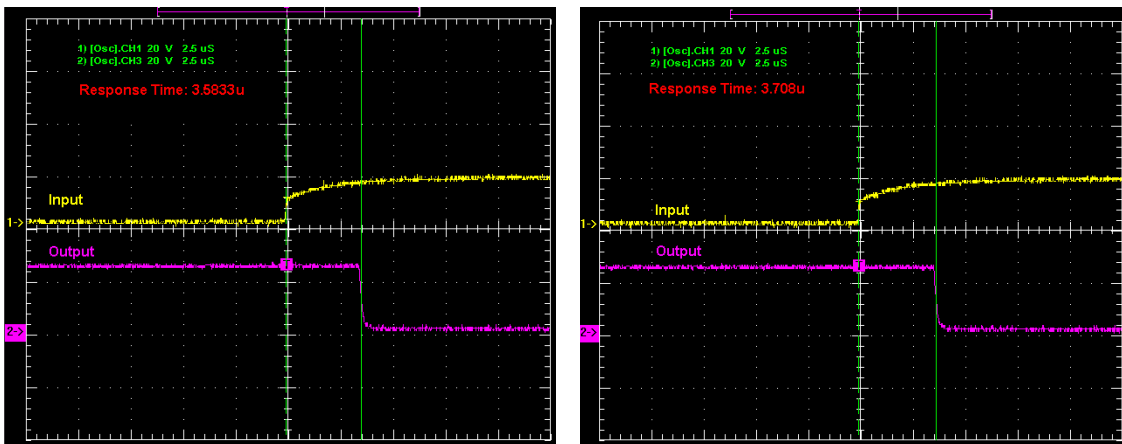


Figure 4.9: On the left side is depicted the response time of an AND gate of two inputs, and on the right side the response time of an AND gate of twelve inputs

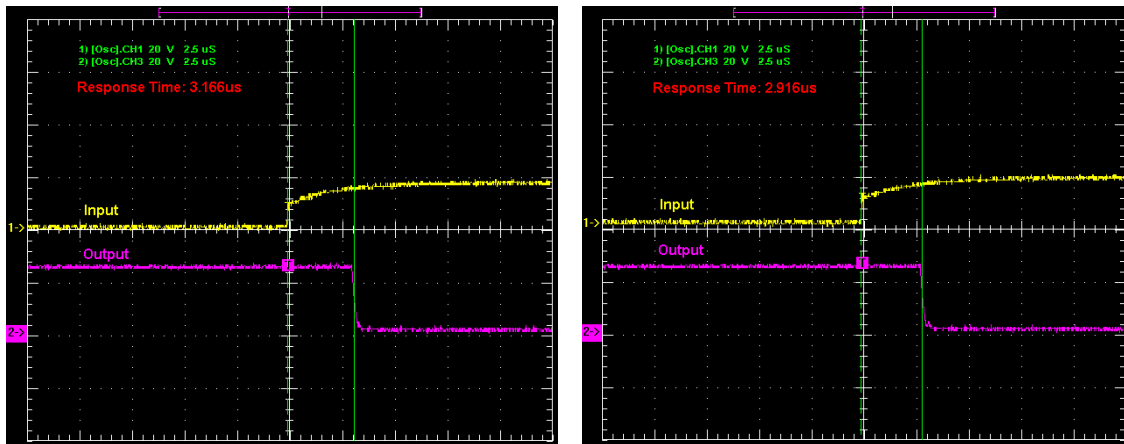


Figure 4.10: On the left side is depicted the response time of an OR gate of two inputs, and on the right side the response time of an OR gate of twelve inputs

If we compare the response time of the OR gate faced to the AND gate we find the OR gate is faster than the AND gate. The explication of this is found in the different inputs combinations used for triggering the output. These combinations affect the input delay introduced by the module that is variable (but always bounded by  $3\mu\text{s}$  as maximum).

- The size of the program (depending on the delays on the inputs and outputs introduced by the module) does not affect the response time since it is executed in parallel and not sequentially. This result has been obtained comparing the response time of a complex program (the Getting started program supplied by Siemens) that occupies a 51% of the FPGA and a very simple one consisting on an OR gate of two inputs that occupies an 11% of the FPGA depicted in Figure 4.11. We have obtained a response time of approximately  $3.5\mu\text{s}$  in the complex program and  $3.2\mu\text{s}$  in the simple one.

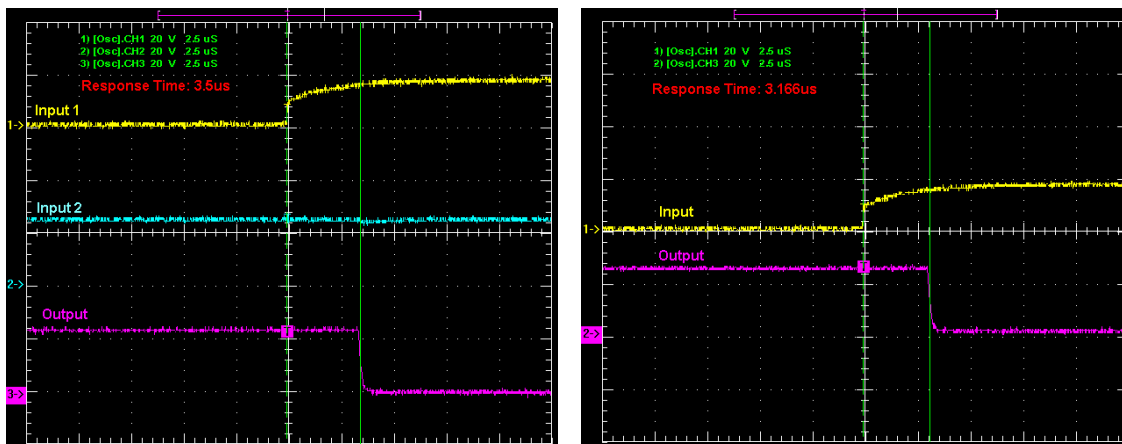


Figure 4.11: On the left side is depicted the response time of complex program (51% of the FPGA), and on the right side is depicted the response time of simple program (11% of the FPGA)

## Evaluation of a fast PLC module in prospect of the LHC beam interlock system

- There is an asymmetry between the activation and deactivation of outputs. The typical response time of output activation is approximately  $3\mu\text{s}$  and the typical response time of output deactivation is approximately  $6\mu\text{s}$ . These results have been obtained comparing the activation and deactivation response time of an OR gate of 12 inputs (see Figure 4.12) verifying Siemens technical specifications.

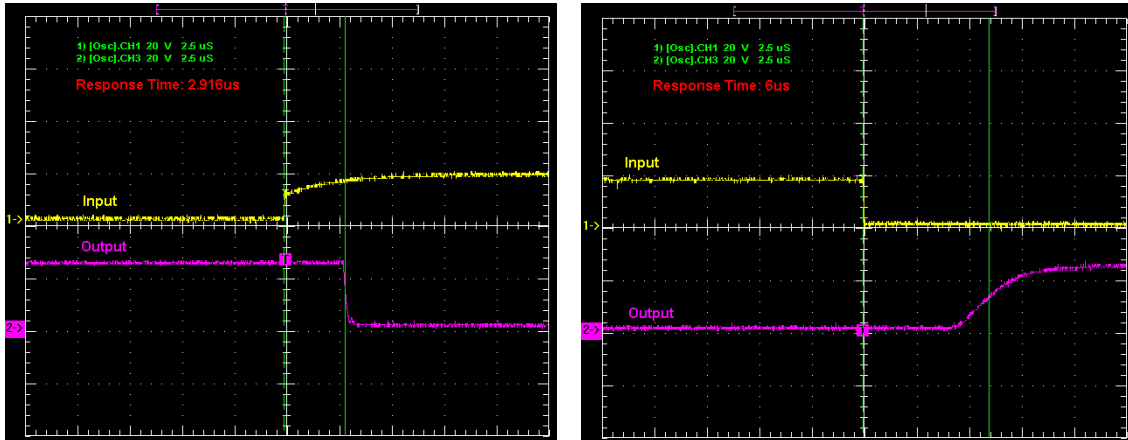


Figure 4.12: On the left side is depicted the output activation response time, and on the right side is depicted the output deactivation response time.

- In the case of using differential inputs we have obtained a different behavior. We have again asymmetry between the activation and deactivation of outputs (Figure 4.13). However, if we compare them faced to the 24VDC general inputs behavior, the differential ones are up to  $2\mu\text{s}$  faster in its activation reaching a response time of approximately  $1.6\mu\text{s}$  (Figure 4.13 left side).

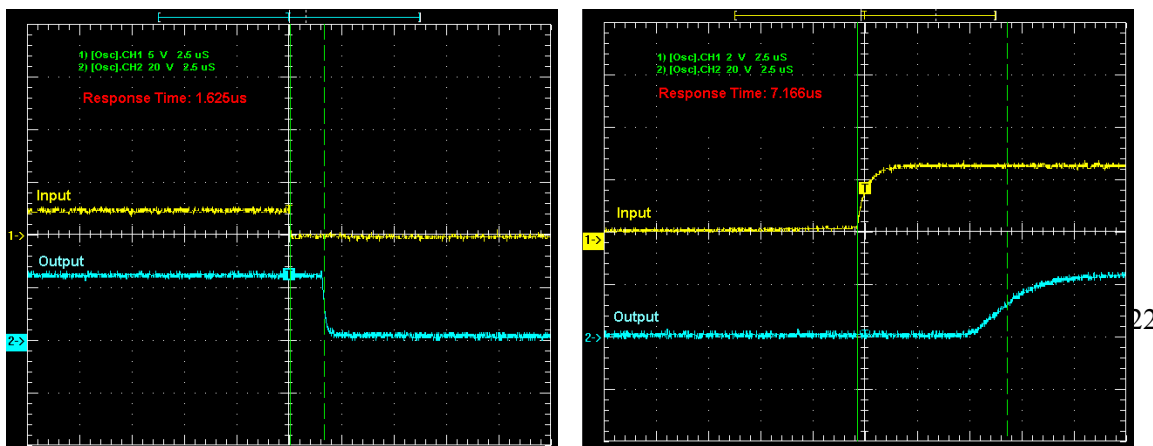


Figure 4.13: On the left side is depicted the output activation response time for differential inputs, and on the right side is depicted the output deactivation response time for differential inputs.

The program executed for Figure 4.13 consists of a single OR gate of three inputs. In the case of the Deactivation the response time is approximately the same than in the case of general 24VDC. In these pictures we have to take into account that the logical input levels are inverted respect to the examples studied for general 24VDC cases.

This corresponds to the technical specifications provided by Siemens. The response time is bounded in the interval  $[2..6]\mu\text{s}$  and this is due to the three general steps needed for the execution of any program inside the FM 352-5 Boolean processor (worst case times): the acquisition:  $3\mu\text{s}$ , the execution of the program:  $1\mu\text{s}$  and finally the Output activation/deactivation:  $2\mu\text{s}$ . Figure 4.14 depicts this conclusion.

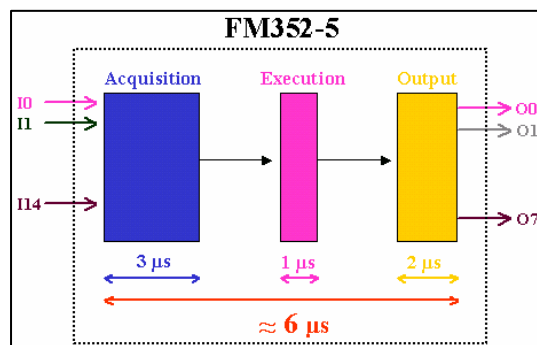


Figure 4.14: General steps for program execution.

- We conclude:
  1. We have a knowledge of the environment  $3\mu\text{s}$  delayed
  2. We could control a process with a granularity of  $6\mu\text{s}$

We have studied also the FPGA occupation of different programs:

- A single logical gate of 2 inputs (AND or OR gate): 11%
- A single logical gate of 12 inputs (AND or OR gate): 12 %
- An AND gate of 12 inputs and an OR gate of 12 inputs: 12%

## CHAINED MODULES CONFIGURATION

Another important study is the evaluation of the response time in chained FM352-5 boolean processors. The reason is that it is possible we have more than 15 inputs from the process to control and monitoring, and a single output. In this case we need at least two chained FM352-5.

We have computed the response time of a chained connection wiring one of the outputs (Q5) of the first module (FM352-5(1)) to one of the inputs (I7) of the second module (FM352-5(2)), measuring the response time from I0 to Q6. Figure 4.15 depicts the connection diagram.

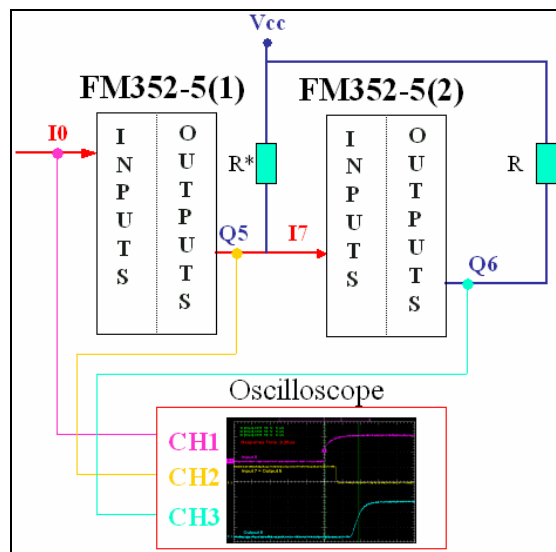


Figure 4.15: Connection diagram for chained modules response time evaluation.

In Figure 4.15 we have to point the different pull-up resistor value for the Q5 output ( $R^*=5k\Omega$  instead of  $10k\Omega$ ), needed for adequate generation of voltage levels for the input I7 with which it is physically connected. The probes of the oscilloscope are connected to the input I0, the output Q5 or input I7 (physically wired) and the output Q6. Then we test the response time for the activation and deactivation of the output Q6 running the control software shown in Figure 4.16.

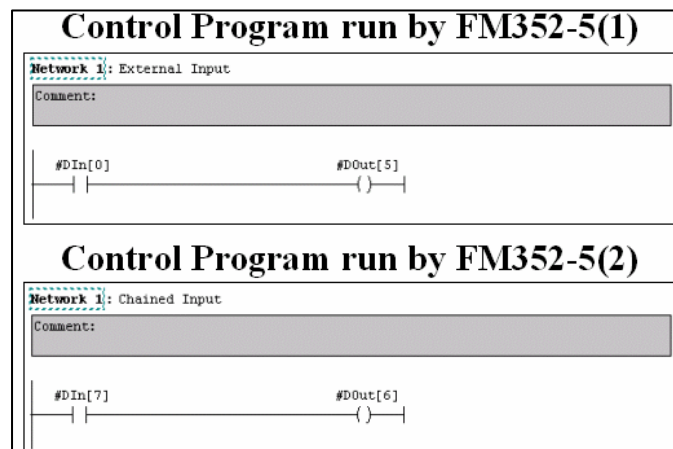




Figure 4.16: Control programs for chained modules response time evaluation.

Running these programs we have checked in first place the response time for the deactivation of output 6 just setting the input 0. The result is depicted in Figure 4.17

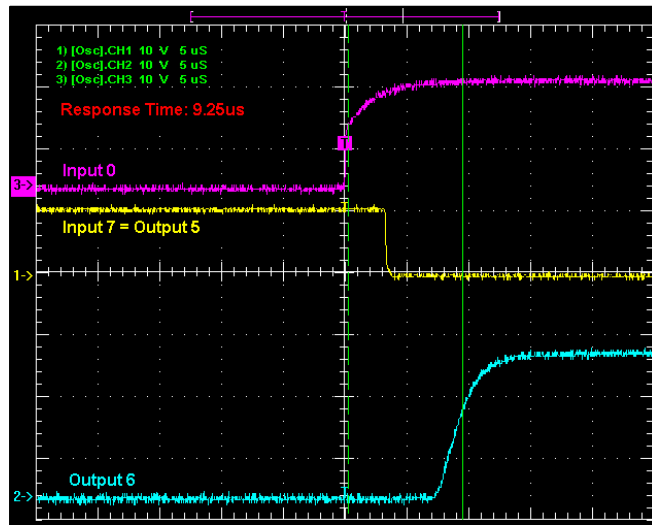
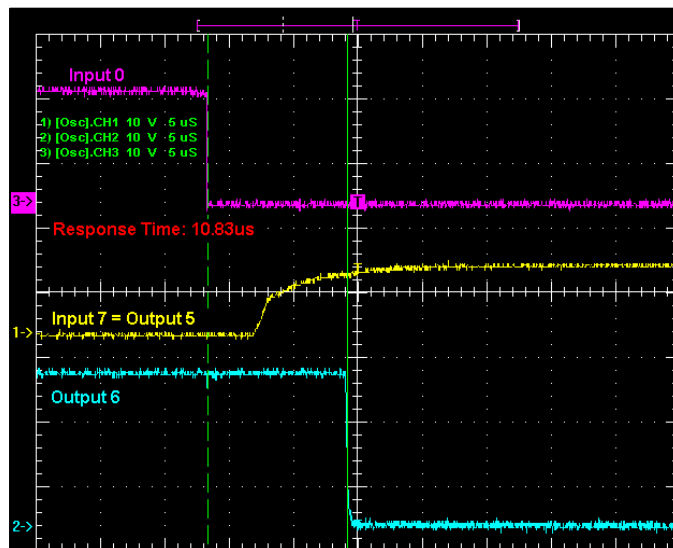


Figure 4.17: Response time evaluation for the output deactivation Q6.

When the input 0 is set, the output 5 is also set (for inputs a logical 1 corresponds to a high voltage level but for outputs a logical 1 corresponds to a low voltage level). Once the output 5 is set (low voltage level), the input 7 has the same voltage (physically wired) but it corresponds to an input deactivation (low level voltage) and then the output 6 is deactivated (high voltage level). The response time obtained is approximately  $9.25\mu\text{s}$  that corresponds to the output 5 activation time (approximately  $3\mu\text{s}$ ) plus the output 6 deactivation time (approximately  $6\mu\text{s}$ ). The rest of the time corresponds to input

We have also tested the output 6 activation 4.18.



the response time of depicted in Figure

Figure 4.18: Response time evaluation for the output activation Q6.

When the input 0 is deactivated, the output 5 is also deactivated (for inputs a logical 0 corresponds to a low voltage level but for outputs a logical 0 corresponds to a high voltage level). Once the output 5 is deactivated (high voltage level), the input 7 has the same voltage (physically wired) but it corresponds to an input activation (high level voltage) and then the output 6 is activated (low voltage level). The response time obtained is approximately  $10.83\mu\text{s}$  that corresponds to the output 5 deactivation time (approximately  $6\mu\text{s}$ ) plus the output 6 activation time (approximately  $3\mu\text{s}$ ). The rest of the time corresponds to input and output delays.

If we compare these results faced to single module response time, we conclude that in the case of the output deactivation it is a less than the double ( $9.25\mu\text{s}$  faced to  $6\mu\text{s}$ ) due to the combination of an output activation and deactivation and the variable input and output delays, while in the case of the output activation it is greater than the double ( $10.83\mu\text{s}$  faced to  $3\mu\text{s}$ ) due to the combination of an output deactivation and activation and the variable input and output delays.

## ***MANAGING MASKS***

Under certain situations input signals should be masked.

The way of applying masks in any CPU-S7 consists of performing logic operations over words or bits of the style AND, OR, or XOR. In the case of the FM352-5 Boolean processor, these operations are restricted to bit logic operations being impossible to apply word masks. Then, if we want to disable some input combination we have to do it bit by bit performing for example an AND operation.

## ***ARMING THE FM 352-5***

In some situations as the module set-up it is interesting to set the module in a known initial or secure state. This process is known as Arming.

In the case of the FM352-5 and our program, we have to independently consider the control part and the monitoring part.

The execution of the control part could be controlled simply using an specific input applied over the AND and OR gates. This specific input could be transmitted from the CPU to the Boolean processor using DBs as described at section 3.3.

The execution of the monitoring part implies the use of frequency divisors. These frequency divisors begin to work when they receive the external clock signal. Then controlling the clock signal we have enough for arming the system. In order to control the clock signal we could apply a mask reading its value from the CPU through a DB as described in sections 3.3 and 4.4. The information storage is also controlled in this way due to signals does not change and then no monitoring information is stored.

## ***ANALYSIS OF EXTERNAL CLOCK SIGNAL GENERATION USING THE FM352-5***

In this section we evaluate the use of a special output signal generated by the FM352-5 for external clock transmission (CLK pin 32).

The purpose of its study is fold:

- Its utility as a failure detection signal for the equipments that compose the LHC testing its presence or its absence.
- Its utility as an external clock generator for counters avoiding the use of additional equipments decrementing the probability of failure.

In order to use it we have to configure an absolute encoder with SSI (Synchronous-Serial Interface) in a master mode. For this purpose we use the parameters configuration tab dialog to set:

- The encoder type: SSI encoder interface
- The clock rate: 125KHz, 250KHz, 500KHz or 1MHz.
- The Delay Time: it depends on the encoder model with which we work. This delay time could be: 16, 32, 48 or 64 $\mu$ s.
- The SSI mode: Master.

The delay time needed in the configuration is used for the synchronization between the FM352-5 module and the encoder, which has always a minimum time delay that must match with the FM352-5 configured value.

We have tested the behaviour of the clock signal generation finding out that due to the delay time the clock pulses are also delayed as depicted in Figure 4.19.

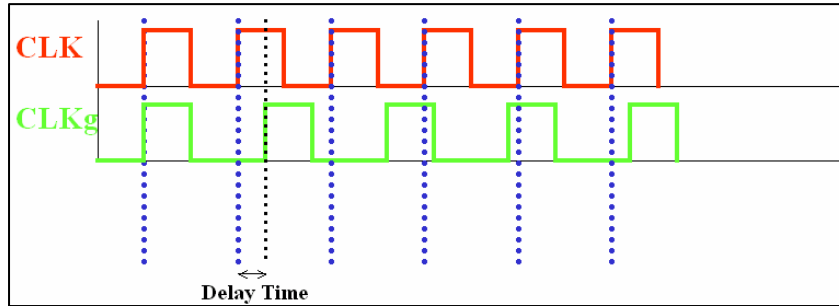


Figure 4.19: Effect of delay time in clock signal generation.

This behavior is not interesting for our purposes due to:

- The frequency absence detector must detect the absence of signal and not a delay in the signal transmission.
- Counters need a square signal with equal half periods.
- We loose a differential input (I12).
- The generated CLK signal is not differential, then if we want to use it for counters we have to use a general input (from the group I0..I11) losing it as a from process input.

## RELIABILITY ISSUES OF THE FM352-5

In this topic we analyze the reliability offered by the FM352-5 boolean processor for its use inside the BIC.

In first place we analyze the diagnostic tools offered by the module. The FM352-5 offers self-diagnostics, output diagnostics, process interrupts, input filters, encoders operating configuration and module diagnostics hardware support enable/disable. These features could be set through the *Parameters Tab* depicted in Figure 4.20.

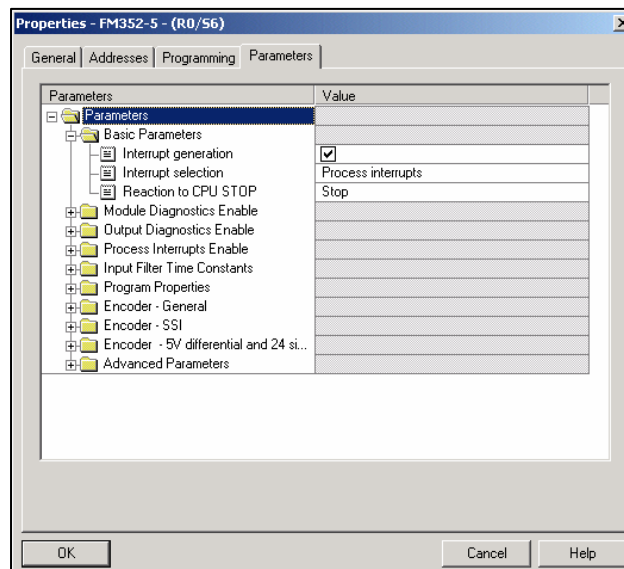


Figure 4.20: Properties tab/Parameters of the FM352-5.

In this way it is possible to configure:

- Module Diagnostics: supply voltages (polarity, low or high voltage, internal fault, etc.), encoders supply faults, MMC diagnostics, etc.
- Output Diagnostics: Overload of the 8 outputs monitoring
- Process Interrupts: report alarm conditions to the CPU, which can trigger an alarm OB (Organization Block) to respond the condition. The module offers up to 8 process interrupts.
- Input Filter Time Constants: for the application of a delay constant to the 15 inputs preventing incorrect input signal readings resulting from noise or other causes.
- Program Properties: how the FM352-5 is to operate in a standalone configuration.
- Encoders: selection of encoders' properties.
- Advanced Parameters: Disable the hardware support of the diagnostics and interrupts provided by the module. In this way it is possible to save memory in our program (related to fixed resources).

In order to measure the reliability of equipments, it is important to know the MTBF (Mean Time Between Failures) tested for that equipment. In the case of the FM352-5 the MTBF tested by Siemens is 19.5 years.

There are also another solutions for the implementation of the heart of the BIC simpler than the FM352-5 but not so flexible and scalable. These solutions are mainly two: the use of a FPGA (as for example a Xilinx FPGA) and the use of a discrete AND or OR chip. We have consulted the MTBF of these components finding it is of the order of thousands of years (for a Xilinx FPGA it could be from 4000 to 25000 years [7]).

These results are logic due to any simple system is always more reliable than a more complex one. The FPGA or the AND or OR gate are very simple components (they are discrete components not inside a PCB) then their reliability must be greater than the FM352-5.

However, the FM352-5 is also a FPGA but combined with another circuits that make it interesting. Figure 4.21 depicts the functional block diagram of the FM352-5 Boolean processor.

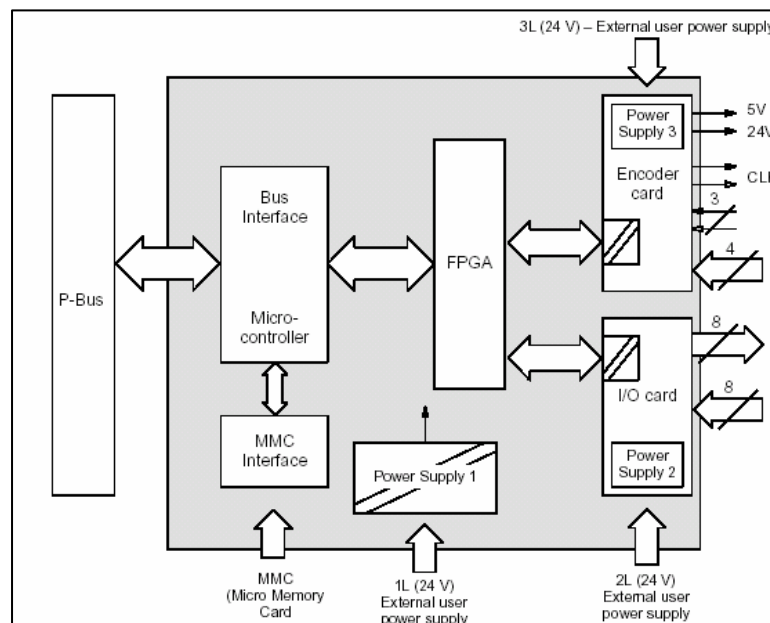


Figure 4.21: Functional Block Diagram of the FM352-5.

The FM352-5 has another components as the encoder and I/O cards, power supply circuits, bus interfaces, and even an EEPROM recorder. All these circuits reduce the global reliability of the system (the system is more complex and the probability of failure is increased) but provide important features from the point of view of its flexibility and scalability:

- Inclusion of the monitoring system of the BIC inside the FM352-5 module.
- Remote access to the module from any network interface (Ethernet, Profibus, RS-232, MPI).
- Remote monitoring and programming of the FM352-5 through the CPU-S7.
- The possibility of failure detection and treatment.
- Fast development and ease of programs implementation using a graphical environment known as STEP7.
- Possibility of saving an MMC and distribute its program between several PLCs.

## **BEAM INTERLOCK SYSTEM: MONITORING AND PROGRAMMING**

The requirements for the beam interlock related to the monitoring applications are:

1. Time Stamp of the trigger (change of any input signal)
2. Timing difference between any two triggered inputs

Then, we evaluate the limitations imposed by the FM352-5 boolean processor for performing both tasks. The solution proposed performs a measurement of time in a relative way and not in an absolute way, that is, we have a time reference. Another problem is how to achieve an absolute time.

The section is structured as follows: firstly we study the programming elements offered by the module, secondly we analyze counters with an external clock generator, next we study how to measure time intervals, in fourth place we analyze several absolute time stamping architectures, next the program identification, and finally the boot of the module from a recorded MMC.

### ***PROGRAMMING ELEMENTS OFFERED BY THE MODULE***

In order to measure the time, the FM352-5 boolean processor offers two primitives: timers and counters. In the case of timers, the programming primitives offered by the module are: Pulse timers (TP16 and TP32), on-delay timers (TON16 and TON32), off-delay timers (TOF16 and TOF32). We have checked them, and

we have found that the time is measured in tenths of microseconds, and then we have a granularity of  $10\mu\text{s}$  for time measurement.

In the case of counters, the programming primitives offered by the module are: up counter (CTU16), down counter (CTD16) and up/down counter (CTUD16 and CTUD32). These counters could work in two ways:

1. Using an internal clock generator through the CP\_GEN programming primitive.
2. Using an external clock generator.

In the first case, we have found out that the CP\_GEN programming primitive is able to generate a signal of 50kHz ( $20\mu\text{s}$  period) as a maximum frequency. The granularity is  $20\mu\text{s}$  for time measuring.

In the second case the manual of the FM352-5 module specifies that the maximum frequency allowed is 200kHz ( $5\mu\text{s}$ ). This feature is analyzed in the next point.

## ***ANALYSING COUNTERS WITH AN EXTERNAL CLOCK GENERATOR FOR TIME INTERVAL MEASUREMENT***

We introduce an external clock signal of 200kHz ( $5\mu\text{s}$ ). This signal must be a square signal with equal half periods. We use the I12 input of the FM352-5 boolean processor. For the introduction of signals as this one, it is interesting the use of 5V differential inputs due to its different asymmetry response time faced to general 24VDC inputs (see Figures 4.12 and 4.13). Then, it is a good idea to use the first 12 inputs as general inputs from the process, and the next 3 ones as special inputs for application-dependant purposes.

We have checked using a TON32 timer if the counts are the double (remember that timers count each  $10\mu\text{s}$  and our counter should count each  $5\mu\text{s}$ ). Our study has demonstrated that it occurs.

The last conclusion about using external clocked timers is that we could measure the time with  $5\mu\text{s}$  granularity. In the next point we relax this constraint using an special programming primitive offered by the FM352-5 boolean processor.

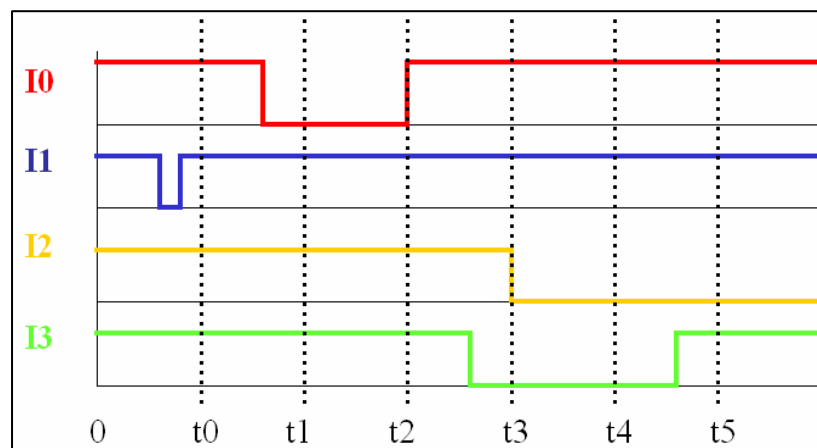


Figure 5.1: Example of time measurement with 4 inputs

Figure 5.1 depicts a possible chronogram of 4 inputs, and the time granularity that we are able to reach with our counter. Any change that occurs between the instant 0 and  $t_0$  for us corresponds to the same instant of time (instant 0), and then the glitch of I1 (if it is detected) has the same time stamp for the change from 1 to 0 and from 0 to 1.

In order to detect the glitch at I1, it must have a duration of at least  $4\mu\text{s}$  in the worst case ( $3\mu\text{s}$  delay of the input plus  $1\mu\text{s}$  for the execution of the program). If we detect this change, we could perform the time stamping that corresponds to instant 0.

## **TIME INTERVAL MEASUREMENT**

The objective is to provide relative time interval measurement between inputs activation. In the next point we analyze different architectures for absolute time stamping acquisition.

In order to perform the time interval measurement for monitoring purposes, we have to analyze which storing data structure is the most interesting, and afterwards how to program the monitoring application.

## **DATA STRUCTURES**

The data structure needed for the monitoring application involves:

- State of all inputs from the process to control (15 per Boolean Processor)
- Time stamping

The next question to solve is the number of bytes needed. For the storage of inputs state we need 2 bytes (15 inputs). In the case of the time stamping we have to analyze different aspects before deciding the number of bytes to use. This analysis is done in the next topics.

## **PROGRAMMING PRIMITIVES**

For the storage of this information, the FM352-5 boolean processor offers two possibilities:

- The use of *Connectors* (similar to marks in an standard CPU-S7) for in memory storage.
- The use of shift registers.



## Evaluation of a fast PLC module in prospect of the LHC beam interlock system

In the case of *connectors* the main problem we have found is they do not allow indirect indexed access. They allow only direct indexed access, that is, we could access to element  $EL[5]$  but we can not access to element  $EL[i]$  where  $i$  is a variable and  $EL$  is a *connector* data type.

The second possibility is the use of shift registers. The FM352-5 offers up to 4 different shift registers:

- SHIFT: implemented in the FB124 for 1 bit input storing from 2 to 4096 bits.
- SHIF2: implemented in the FB125 for 2 bits input storing from 2 to 2048 pairs of bits.
- SHIF4: implemented in the FB126 for 4 bits input storing from 2 to 1024 nibbles.
- SHIF8: implemented in the FB127 for 8 bits input storing from 2 to 512 bytes.

The maximum number of shift registers we could use is 10, then the maximum storage capacity corresponds to 5Kbytes (10 SHIF8 registers).

Besides the data input, shift registers have two interesting control inputs:

- Reset: when the shift register is enabled and this input is set, the whole register (all its states) is set to 0.
- Clock: when the clock input transitions from 0 to 1, the value of the data input is shifted into the first stage of the shift register, and is shifted for each subsequent clock edge. The output is set by the last position in the shift register.

The most elegant decision consists of the choice of shift registers, storing the information each time any input signal changes. In order to detect a signal change, we perform a XOR (eXclusive OR) between the current inputs state and the previous input state (inputs state in the previous scan cycle of the boolean processor).

For the implementation of this signal detection we have use two SHIF8 registers with a length of two bytes that store the previous state of all the inputs (15 per Boolean processor). Each scan cycle our program performs an XOR between the current inputs ( $I1..I15$ ) and the corresponding previous inputs state ( $I1\_old..I15\_old$  see Figure 5.2) generating the Save signal for information storing. The boolean processor does not provide the XOR gate in LAD (LADder logic programming language) then we have implemented it using AND, OR and NOT gates.

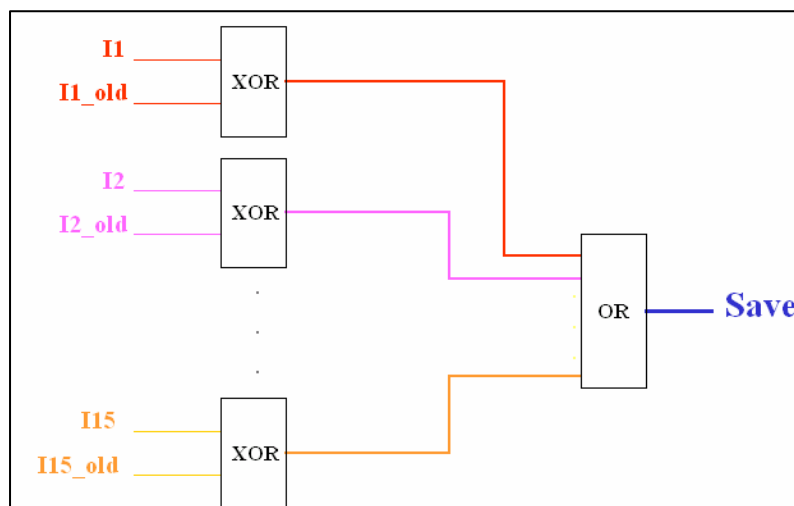


Figure 5.2: Inputs' changes detection

The following problem to analyze is the programming primitives for time measurement. In sections 5.1 and 5.2 we have discussed which was the best alternative concluding that the most interesting one is the use of counters with an external clock source providing a granularity of  $5\mu\text{s}$ .

The counters primitives offered by the FM352-5 module are 16 bits counters (CTU16, CTD16 and CTUD16) and 32 bits counters (CTUD32). In order to select the length we have to take into account the overflow of these counters depicted in Table 5.1.

	MAX.COUNT	TIME
16 Bits Counter	32767	0.16 seconds
32 Bits Counter	2147483647	2.98 hours

Table 5.1: Maximum count value and corresponding time

In Table 5.1 we have considered that counters are incremented each  $5\mu\text{s}$  for overflow considerations. If we take into account these results, we need an additional counter to increase the value of a measured interval. It counts the number of overflows of the reference counter. Figure 5.3 depicts the connection between both counters for time interval measurement.

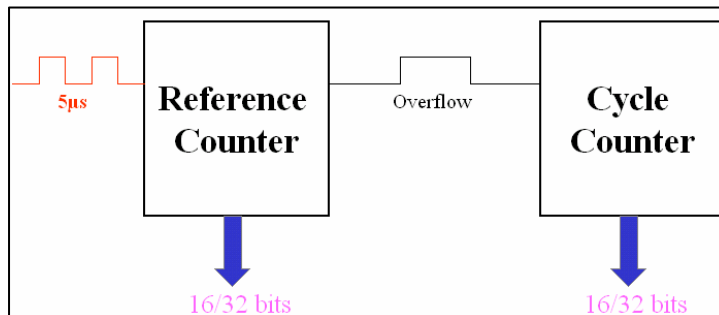


Figure 5.3: Interval measurement using two counters

	Time Measurement
Both counters of 16 Bits	1.49 hours
One counter of 16 bits and the other counter of 32 bits	11.15 years
Both counters of 32 Bits	7311.78 centuries

Table 5.2: Maximum measurable Time

The main problem we have found in the use of counters is that the FM352-5 module only works with INT and DINT data types for the storage of the current count value, however Shift Registers need a Binary input.

Initially this problem is easy to solve converting the INT or DINT to Binary, however this conversion cannot be done with the limited instruction set of the FM352-5 module and the 12 phases limitation per scan cycle. We have checked several alternatives for INT to Binary conversion:

- Performing 16 divisions by two storing the rests and the final cocient. The problem found is that we have as a maximum 12 sequential phases per scan cycle.
- Division of the INT number in two bytes performing 8 divisions in parallel (8 phases per byte). The problem found is that the module cannot work with the Byte data type.
- Use of word masks: we divide the INT number in two bytes applying word masks and performing the 8 divisions in parallel. The problem found is that the module has not word logic operations typical of a standard CPU-S7.
- Consult to *Siemens* provider which answer is the impossibility to do it.

The solution we have finally adopted is the implementation of our own time interval measurer using the limited instruction and primitives set of the FM352-5. with this purpose we have checked two alternatives:

- Implementation of a counter using flip-flops. The problem we found is that the module only allows working with RS asynchronous flip-flops which introduce a great complexity in the method.
- Use of frequency divisors. This is the solution finally adopted and discussed in topic 5.3.3.

Once decided the programming primitives for time interval measurement, we have to define its length in bits. We consider that a period of a week is enough, then considering increments each  $5\mu\text{s}$  the maximum number of bits needed are 37 bits or 5 bytes obtained through the expression:  $7*24*60*60*1000*1000 / 5\mu\text{s}$ .

We conclude to store inside the FM352-5 module a total of 7 bytes per monitoring record:

- 2 bytes for the state of the inputs: we have up to 15 inputs per module.
- 5 bytes for the Time Interval measurement

## USING FREQUENCY DIVISORS

In this topic we present the solution adopted for time interval measurement. Our solution is based on the use of the FM352-5 programming primitive *BiScale* (Binary Scaler) implemented in the FB112.

The *BiScale* provides a way to produce a series of output pulses at half the rate of the input pulses. It has an input known as *C* and an output *Q* both boolean. Each rising edge at input *C* inverts the output *Q*, effectively dividing the frequency of the input by two, as shown in Figure 5.4.

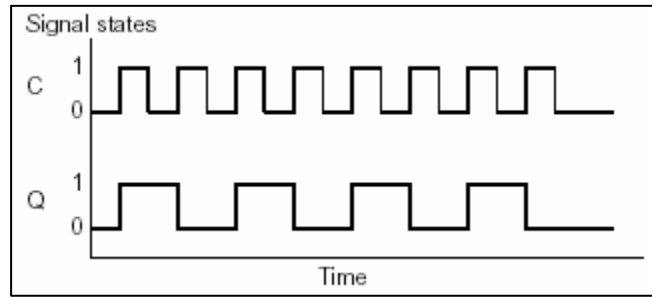


Figure 5.4: Timing diagram for *BiScale*

Then, the implementation of the counter using this primitive is initially simple. We use 35 frequency divisors (7 per byte) synchronized by the external clock signal of 200KHz (see Figure 5.5). The problem we found is that the boolean processor only supports 11 nested binary scalars.

The solution of this problem is to divide in bytes the counter nesting only up to 7 *BiScalers* per byte as a maximum. We manually divide the frequency of the last bit of each byte feeding the next byte. This manually division consist of the use of a two bits counter which is incremented by the up-edge of the clock signal, when the counter is 1 we maintain a 1 at the output and when it is 2 we put an 0 at the output and reset the counter. This output is used as the first bit of the next byte.

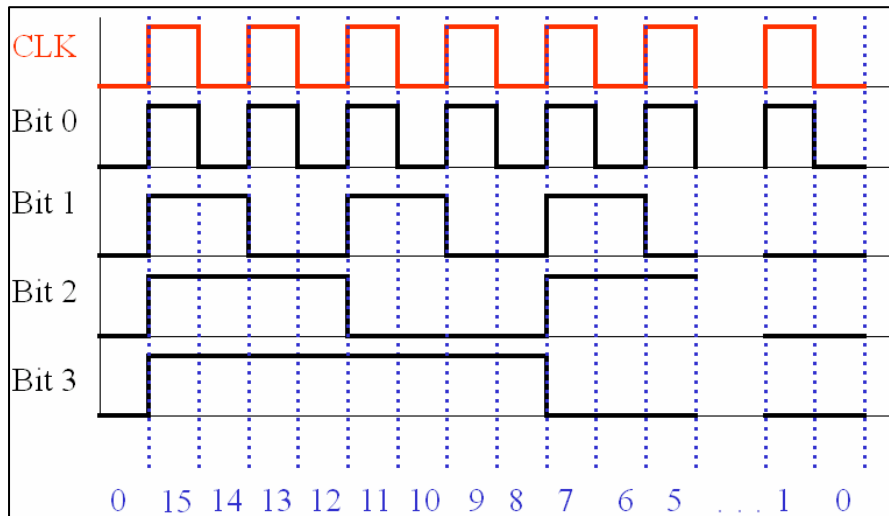


Figure 5.5: Example of a down counter of 4 bits using *BiScalers*

The use of frequency divisors as time interval measurer provides an interesting feature, with an external clock generator of 200KHz (5 $\mu$ s period), we are able to count at a frequency of 400KHz (2.5 $\mu$ s period). This is very interesting from the point of view of time interval measurement granularity due to we could

measure minimum time differences of up to  $2.5\mu\text{s}$ . We have checked this comparing the behavior of a standard counter and our frequency divisor using both an external clock signal of 200KHz. The result is that our frequency divisor counts at the double speed than the standard counter.

It is important to stress that using the frequency divisors, some error in the time measurement occurs due to input delays. However this error is of the order of  $\text{ns}$  being negligible for our purposes.

## DOUBLE BUFFERING TECHNIQUE AND ITS FM352-5 IMPLEMENTATION

Finally, we analyze the programming technique for the monitoring application. On one hand we have to write inside the FM352-5 boolean processor the records explained above, and on the other hand, the CPU S7 have to read these records. The FM352-5 boolean processor writes the information each  $4\mu\text{s}$  (worst case), however the CPU S7 is able to read these records only each  $5000\mu\text{s}$  (worst case). Besides, the CPU S7 could only read two records (assuming that the size of each record is 7bytes) per read. Then, the problem here is the adaptation of the very different rates of reading and writing of each device.

We have chosen the technique known as *double buffering* (see Figure 5.6). It consists of two buffers we have named as *memory buffer* and *interface buffer*. The *memory buffer* is filled each  $4\mu\text{s}$  (as a minimum period), and each time the CPU requires a monitoring record (5ms as minimum period), the *interface buffer* stores the new information of the *memory buffer* making a copy. In this way, the *memory buffer* has always the most updated information, and the CPU reads from the *interface buffer* at its own rate (much more slow than the writing rate).

Afterwards, the CPU S7 could send the monitoring information to any other PC/PLC via IE, PROFIBUS, MPI, RS232 or any other out-world interface.

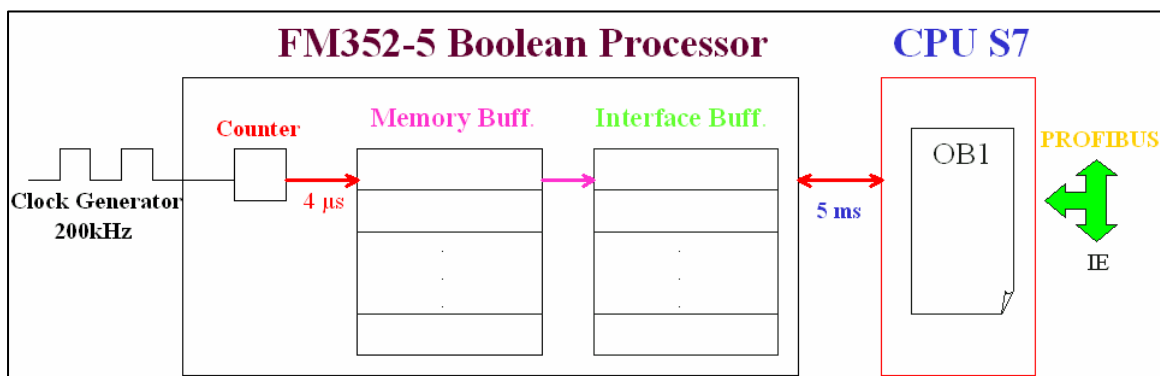


Figure 5.6: Double buffering technique

The limitations of the application of this technique are the next ones:

- The very low bandwidth communication between the CPU-S7 and the FM352-5 boolean processor: 14 bytes each  $5000\mu\text{s}$  (only 2 records/access).
- The FM352-5 does not allow working with pointers, and then the management of both buffers is complex from the programming point of view (remember the limited instructions set of the boolean processor).

## Evaluation of a fast PLC module in prospect of the LHC beam interlock system

- The FM352-5 does not allow the access to the internal records stored in shift registers, that is, we could access only to the outputs that correspond to the first information stored.
- The maximum memory size for information storing inside shift registers is 5kbytes.

Given the previous limitations, we propose the next way for the implementation of the double buffering technique inside the boolean processor depicted in Figure 5.7.

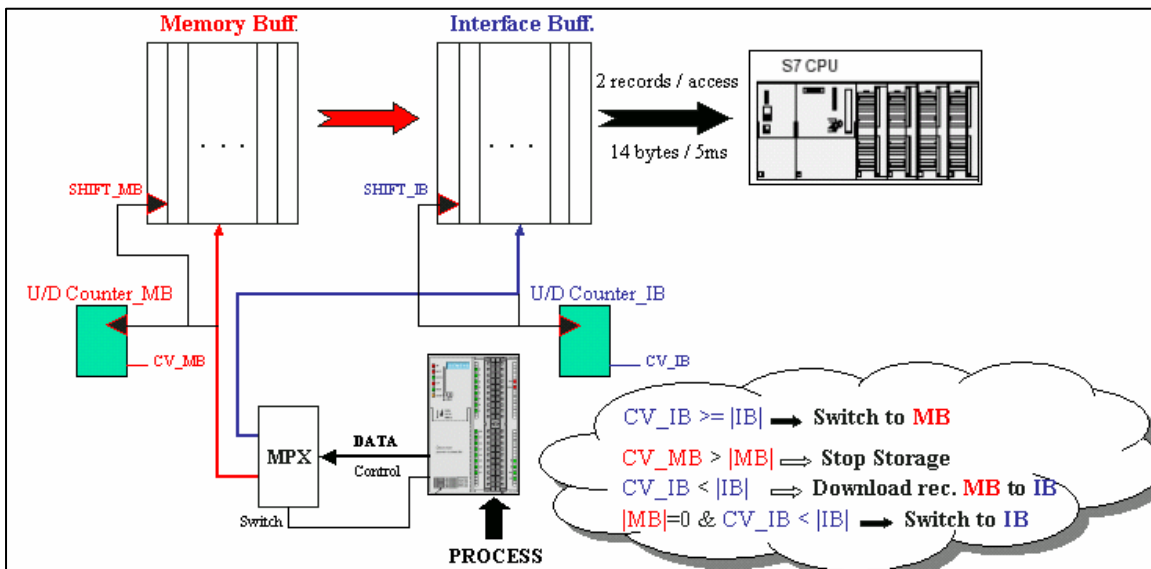


Figure 5.7: Implementing the Double Buffering Technique inside the FM352-5

The FM352-5 module usually writes the information inside the *interface buffer* from where the CPU-S7 reads it. When the input signals begin to change with a period minor than 5ms, it implies the possibility of losing information. In order to avoid it, the *MPX* switches (when the *interface buffer* is full) and all the information from the FM352-5 begins to be stored inside the *memory Buffer*. It is possible that the *memory buffer* overflows. In order to control the free space inside the memory buffer we use an up/down counter known as *counter\_MB*, which is incremented each time a new record is stored and decremented each time

a record is read. In the case that the current value of the *counter\_MB* reaches the maximum capacity of the *memory buffer* we begin to lose information then we have to stop the storage, but we have been able to store up to 73 records considering a 512bytes *memory buffer*. It is important to notice that in this case we are losing the most updated information but not the historical information, which is the most useful information for postmortem analysis.

There is also another up/down counter associated to the *interface buffer* known as *counter\_IB*. This counter is incremented each time a new record is saved inside the buffer and decremented each time a record is read. When the current value of the *counter\_IB* is minor than the size of the *interface buffer*, it implies that we have space inside the *interface buffer* for a new record that is read from the *memory buffer*.

When the *memory buffer* is empty and there is space inside the *interface buffer*, the MPX switches and all the information begin to be stored inside the *interface buffer*.

It is important to stress that at any moment, the CPU-S7 is reading only from the *interface buffer* at its own rate (14 bytes each 5ms) imposed by the FM352-5 to CPU communication pipe.

The main advantage of this implementation is that we do not lose information unless the *memory buffer* is full. If we had only one buffer for information storage, we would lose much more information when the input signals began to change with a period minor than 5ms between changes.

## **ABSOLUTE TIME STAMPING ARCHITECTURES**

In this section we provide a set of architectures commercially available that address the problem of absolute time stamping. Up to now we have performed time stamping in a relative way taking a clock reference, but it is required to know with the highest accuracy the absolute time in which the events had happened for the postmortem analysis.

The first architecture is based on a PROFIBUS DP Master/Slave configuration (see Figure 5.8). Several equipments compose it:

- GPS (Global Positioning System) receiver: for time acquisition.
- Siclock module: *Siemens* module for time distribution.
- PLC *Siemens* S7-400: CPU S7-400, Ethernet CP343-1 module and a CP 342-5 Profibus DP module.
- ET200 *Siemens* equipment: IM153-1 Profibus DP slave module, special DI/O module, and the FM352-5 Boolean Processor.

Using this architecture, we are able to obtain absolute time stamping with millisecond precision.

The working of the system is as follows: the Siclock module receives from the GPS this UTC (Coordinated Universal Time) over RS-485, it then generates a Sinec H1 packet and sends it over the Ethernet, then the PLCs set the local clock with this time. In this way, the Siclock module distributes the time through the industrial Ethernet network segment with millisecond precision.

## Evaluation of a fast PLC module in prospect of the LHC beam interlock system

The Profibus DP slave (IM 153-1) located in the ET200 equipment could obtain the absolute time stamp by request to the Profibus master DP through the Profibus network. In the ET200 equipment resides also a specific DI/O module and the boolean processor. When an input signal (I1..I15) is set, it is received by the DI/O module and the FM352-5. The FM352-5 performs its monitoring and control tasks while the DI/O module interrupts the IM153-1 which requests the absolute time stamp.

It is important to stress that we only need this absolute time stamp at the beginning of FM352-5 working due to we have relative time stamps inside the FM352-5 and doing a simple addition we could obtain the rest of the absolute time stamping values.

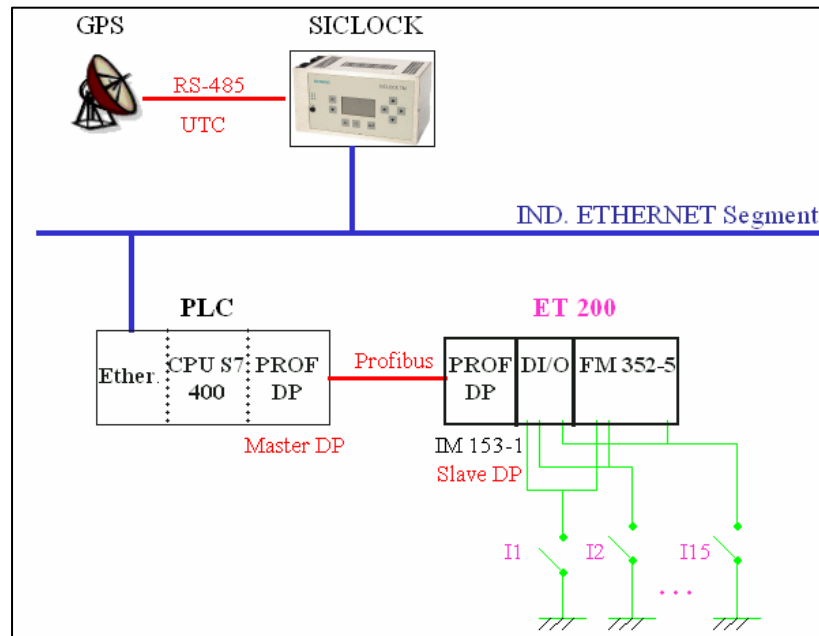


Figure 5.8: Absolute time stamping architecture using Siclock and Profibus.

The main drawbacks of this technique are:

- It is mandatory the PLC CPU is a S7-400.
- We could only achieve millisecond precision for absolute time stamping.
- An Industrial Ethernet network is not a Real-Time network then we cannot ensure time bounds for absolute time stamping.
- The time format is UTC and not IRIG-B (Inter-Range Instrumentation Group).



## Evaluation of a fast PLC module in prospect of the LHC beam interlock system

- Relative to the communication between the CPU-S7 and the Boolean processor, the update time of the FM352-5 in a Profibus configuration is slower than in a central configuration due to it will be 5ms plus the runtime of Profibus connection.

Several solutions that could relax the above problems are the next future developments:

- PLC CPU as S7-300. *Siemens* is currently working on it.
- Siclock working with specific time management protocols as (S)NTP (Network Time Protocol).
- Siclock working with Profibus networks (it currently offers RS-485 interface). In this way we have “Real-Time” performance.
- Other architectures for absolute time stamping using IRIG-B and the FM352-5 in a remote or central configuration.

The next architecture we present is based on IRIG-B protocol [4] and allows the use of Profibus as Real-Time communications network with a CPU S7-300 (see Figure 5.9).

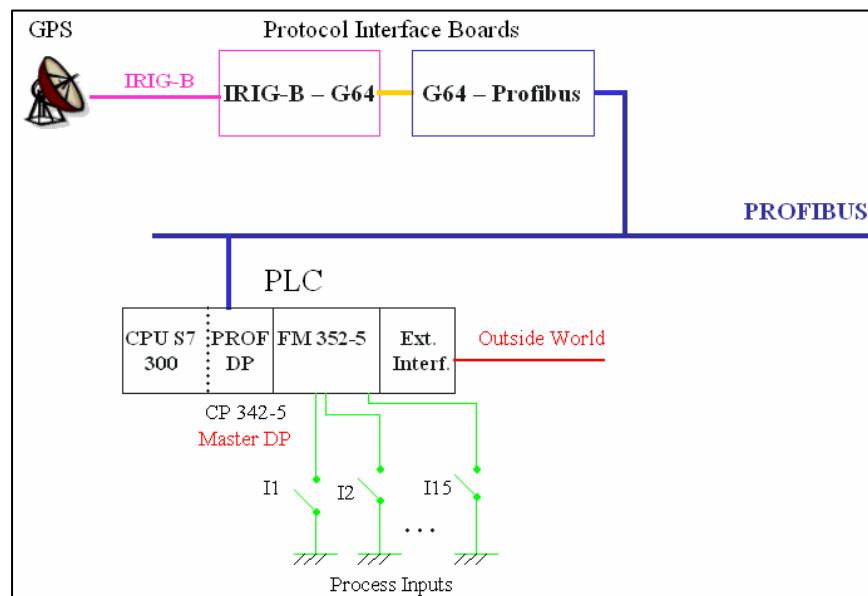


Figure 5.9: Absolute time stamping architecture using IRIG-B/G64/Profibus

In this architecture the GPS receiver communicates the absolute time coded in IRIG-B to an IRIG-B/G64 interface board [5], this board is connected to a G64/Profibus interface board [6] that finally is connected to the PLC through the Profibus DP interface module.

The main advantages of this architecture are:

- The timing precision only depends on the GPS receiver and Profibus communications.

## Evaluation of a fast PLC module in prospect of the LHC beam interlock system

- It works with a “Real-Time” communications network for time distribution.

The main drawback is its reliability being reduced by the number of different equipments used.

It is possible to modify this architecture in which the FM352-5 works as a Profibus slave, however its performance is worse due to communications delay with the CPU-S7 introduced by the Profibus connection.

The next architecture we present is based on an IRIG-B decoder board that provides absolute time stamping to the CPU S7-300 through digital lines connection (see Figure 5.10).

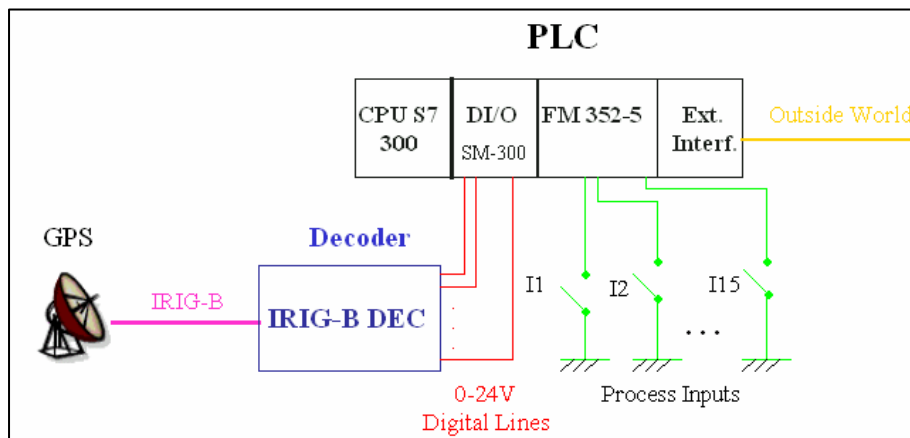


Figure 5.10: Absolute time stamping architecture using IRIG-B Decoder

In this architecture we have an IRIG-B decoder board which function is to provide absolute time stamping to the PLC through a DI/O PLC module (SM-300 family). The interface board converts IRIG-B frames to digital 0-24 lines that are read by the PLC CPU.

The advantages of this architecture are its simplicity (ease of programming and connection) and reliability (we have only one equipment for absolute time provision to the CPU).

The main drawbacks are:

- It is a point-to-point communication. It is not possible the time distribution between several PLCs.

## Evaluation of a fast PLC module in prospect of the LHC beam interlock system

- The digital lines must be short lines for electronic reasons then the absolute time provision system must be physically near from the PLC.
- We loose the protection features provided by an industrial network such as Profibus in the presence of electromagnetic radiation.

Another architecture we present is based on an IRIG-B to RS232C board that provides absolute time stamping to the CPU S7-300 through the RS-232C PLC module (see Figure 5.11).

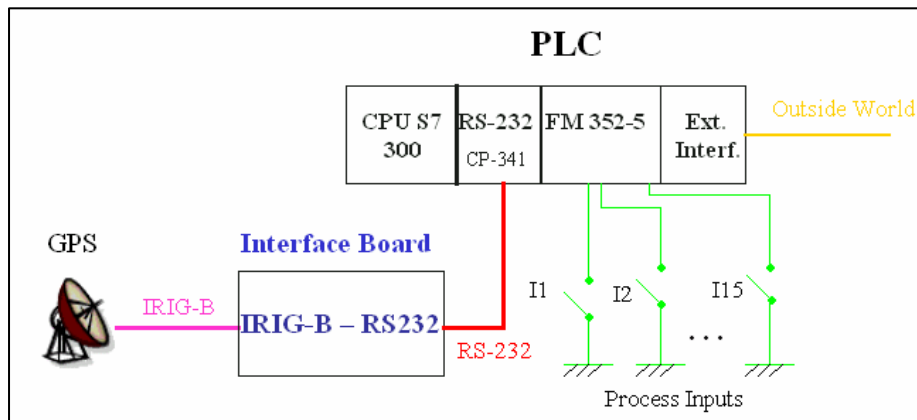


Figure 5.11: Absolute time stamping architecture using IRIG-B/RS232C

In this architecture we have an IRIG-B to RS-232C board which function is to provide absolute time stamping to the PLC through the RS-232C PLC module (CP-341). The interface board converts IRIG-B frames to RS-232 signals that are read by the PLC CPU.

The advantages and drawbacks of this architecture are similar to the previous one: simplicity and reliability but point-to-point communication, physically near to the PLC and protection less in the presence of electromagnetic radiation (RS-232 link). An additional drawback of this architecture is the limitation of baud rate communication imposed by the RS-232C standard of 19200 bauds.

There is a final architecture based on NTP and a high accuracy pulse generator, which provide absolute time stamp with millisecond precision. It is depicted in Figure 5.12.

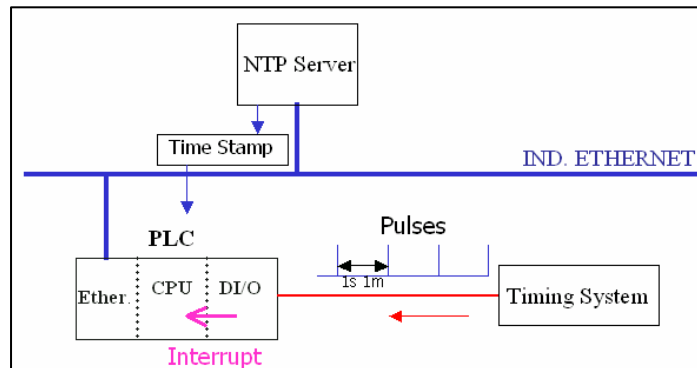


Figure 5.12: Absolute time stamping architecture using NTP & pulse generator.

In this architecture we find a NTP server which target is the provision of absolute time to PLC connected through an Ethernet Network. The other component is the Timing System which target is the generation of a high accuracy pulse per second or per minute.

Given these components, the system works as follows: The PLCs connected via the Ethernet network receive the Time from the NTP server (all the PLCs sooner or later receive the same value due to the broadcast operating mode of an Ethernet network) and also a high accuracy pulse from the timing system each second or minute. When the pulse arrives, an interrupt from the DI/O interface to the PLC CPU is generated, in this way the PLCs could synchronize their clock with the NTP distributed time using the pulse interrupt as reference.

The main advantages of this architecture are their wide applicability to any PLC (Siemens S7/400, S7/300, Schneider Premium and Quantum), its simplicity, and its low price. Another important advantage is the possibility of multiple synchronization of several PLCs connected to the same network independently of the number of segments or intermediate equipments.

The main drawback of this architecture is the unpredictability introduced by the Ethernet network, but easily overcome using Profibus or any other Real-Time network. It is also important to point that the time required for the DI/O interrupt module must be bounded in order to define the synchronization precision being a good candidate the FM352-5 (microsecond scan cycle).

## **PROGRAM IDENTIFICATION**

It is important to know which program is being executed inside the FM352-5 on operation, obtaining this information through the CPU S7 PLC even remotely.

There are two ways of doing this. The first one is by direct access to the FM352-5. Each time we load a program in the FM352-5, the PLC CPU sends a PIN (Program Identification Number) to the FM352-5 module, which stores it in its memory. Then, at any time we could know the running program through the CPU, reading this memory position from the FM352-5. For the implementation we have used a field known as PIN of the *Connector* structure. It is a WORD (16 bits) data type. In the DB5.PIN we put a program identification (PIN) that the CPU sends to the FM352-5, for example W#16#61A8. If the CPU wants to read the PIN of a running program, it has to consult the DB6.PIN field where the FM352-5 sends its PIN.

The second way consists of accessing the CPU reading the FB (Function Block) that the FM352-5 is being executed. In this way we could know the PIN and also the source code being executed in the FM352-5.

Both ways could be done remotely through a Profibus network, an Ethernet network or any other network interface.

## ***BOOTING THE FM352-5 WITH A PROGRAMMED MMC***

We discuss the measurement of the time needed by the FM352-5 module with a programmed MMC to begin its execution from a power failure.

In order to measure this time, we have written a program in the CPU-S7 and in the Boolean processor for measuring the time from the instant the CPU begins to work to the instant the FM352-5 module boot.

The program in the CPU-S7 simple starts a timer that is stopped when the FM352-5 module boots. For this test it is needed to save the FM352-5 inside a MMC.

The sequence of steps followed is: firstly we load the CPU-S7 program inside the CPU maintaining its switch selector in the RUN mode, afterwards we insert the programmed MMC inside the FM352-5 slot maintaining the switch position in the RUN mode, next we power off the PLC and power it on again, and finally we wait for the time needed by the FM352-5 module for reading the program from the MMC. This time measured by the CPU-S7 is approximately 65 seconds for our monitoring and control program described in previous sections with a FPGA percentage of occupation of 49%.

## **FURTHER WORK**

This note details the evaluation work of the FM352-5 boolean processor. For the LHC beam interlock system, a VME based solution has been chosen. This platform currently offers the proper monitoring and control tools required by this application. However, the FM352-5 module could be a good candidate for another applications which require fast processing under PLC based architectures.

## ACKNOWLEDGEMENTS

The author likes to thank numerous colleagues that have been involved in the discussions. In particular I strongly appreciate the input from Bruno Puccio and Rudiger Schmidt. It is difficult to find more proficient people about this matter.

## REFERENCES

- [1] F. Bordry, R.Denz, K-H.Mess, B.Puccio, F.Rodriguez-Mateos and R.Schmidt, *Machine protection for the LHC: Architecture of the Beam and the Powering Interlock systems*, LHC Project Report 521, 2001.
- [2] Vicente Soriano, Manuel Zaera, Carlos Palau, Manuel Esteve, *Comunicaciones Industriales. Programacion de PLCs y PROFIBUS* (in Spanish), ISBN 84-699-3375-2, 2000.
- [3] Siemens, *FM352-5 High-Speed Boolean Processor. User Manual*, Edition 12/2001.
- [4] Benjamin Todd, *IRIG/AFNOR Serial Time Code Overview*, CERN SL-CO report, August 2002.
- [5] Christoph Knaupp, *Development of a serial time code receiver using the IRIG standard*, CERN SL-CO/TI report, Spring 2001.
- [6] E. Carlier et al, *Profibus-DP to G64 Configurable Interface*, SL-Note-2001-016 BT, 6 April 2001.
- [7] <http://www.xilinx.com>