JANUS -
General Telemetry Test Transmitter

Robert Rönnholm
Abstract

The purpose of this thesis was to develop a device that transmits telemetry data. The request for such a device was made by RFN – a government aeronautic test range located near Vidsel, Sweden. The data sent out from the device will consist of a simulation of a customer’s test flight transmission.

One of RFN’s demands was that the PCM bit rate of the device would reach 20 Mbit/s. This prohibited the mere use of a PC. Instead, an electronic front-end was designed and integrated with a PC104, which would contain a web based user interface as well as serve as real-time sensor reader. These real-time sensors included a GPS receiver and a flight termination system.

Another demand was that the system would be tough enough to be carried under a helicopter or towed by a plane, all seasons of the year. Considering this, the PC104 all electronics had to be rugged and temperature resistant.

The design of the front-end proved to be a challenging task. The sought for wide tuning possibilities of the finished product called for an advanced schematic, with 4 different RAM groups and more than 40 other IC’s.

With the current prototype it is possible to set the output bit rate, the data alignment and the number of bits per word, together with other traits of the transmitted data matrix.

After the development time and a revision of the project goals, this project was a success.
Preface

This master thesis was carried through for the missile test range RFN in Vidsel, from December 2005 to February 2007. The major part of the work has been done at EISLAB, at the Department of Computer Science and Electrical Engineering, Luleå University of Technology.

I would like to thank:

- Jerry Lindblom, without whose help, this project would not have been what it is today.
- Ingemar Nyström at RFN, for development aid, report feedback and employment connection during the thesis.

Robert Rönnholm
A clear-skied Luleå
2007-02-05
Contents

1. INTRODUCTION .............................................................................................................. 4
   1.1. BACKGROUND ............................................................................................................. 4
       1.1.1. RFN ......................................................................................................................... 4
       1.1.2. Telemetry................................................................................................................ 5
   1.2. OBJECTIVES ............................................................................................................ 6

2. SPECIFICATIONS .......................................................................................................... 8
   2.1. INITIAL REQUESTS FROM RFN .................................................................................. 8
   2.2. REQUIRED SPECIFICATIONS ................................................................................. 8
   2.3. CONSIDERING PROJECT SPECIFICATIONS ........................................................... 8

3. SYSTEM DESCRIPTION .................................................................................................. 9
   3.1. SYSTEM ARCHITECTURE ........................................................................................ 9
   3.2. A VIRTUAL ASPECT ................................................................................................ 11
   3.3. CONCERNING THE SYSTEM DESCRIPTION ........................................................... 11

4. HARDWARE .................................................................................................................. 12
   4.1. PC104 ....................................................................................................................... 12
   4.2. GPS ........................................................................................................................... 13
   4.3. FTS ............................................................................................................................ 13
   4.4. ELECTRONIC FRON-END ....................................................................................... 14
   4.5. TRANSMITTERS ....................................................................................................... 16
   4.6. POWER SUPPLY ...................................................................................................... 16
   4.7. HARDWARE CONCLUSIONS ................................................................................ 16

5. SOFTWARE ................................................................................................................... 17
   5.1. OPERATING SYSTEM .............................................................................................. 17
   5.2. WEB SERVER ........................................................................................................... 17
   5.3. USER INTERFACE ................................................................................................... 17
       5.3.1. The user interface revisited ................................................................................... 18
   5.4. HARDWARE CONTROL PROGRAMS ....................................................................... 19

6. THE FLEXIBILITY OF THE JANUS SYSTEM .................................................................. 20
   6.1. TELEMETRY FRAME INTRODUCTION .................................................................. 20
   6.2. FRAME SYNCHRONIZATION PATTERN ................................................................. 20
   6.3. SUBFRAME COUNTER ............................................................................................ 21
   6.4. DATA WORDS .......................................................................................................... 21
   6.5. DYNAMIC WORDS ................................................................................................ 21
   6.6. REAL-TIME DATA ................................................................................................... 23
   6.7. OTHER SIMULATION FEATURES OF THE JANUS .................................................. 24
   6.8. FLEXIBILITY CONCLUSIONS ................................................................................ 24

7. CONSTRUCTION ............................................................................................................. 25
   7.1. WIRED TEST CARDS .............................................................................................. 25
   7.2. CIRCUIT CARDS ...................................................................................................... 25
   7.3. CONSTRUCTION EQUIPMENT AND SOFTWARE ................................................... 26

8. EVALUATION ................................................................................................................ 27
   8.1. WIRED PROTOTYPE .................................................................................................. 27
       8.1.1. Tested prototype parts ........................................................................................ 27
       8.1.2. Omitted prototype parts ...................................................................................... 28
       8.1.3. Test equipment .................................................................................................. 28
   8.2. PROJECT PROGRESS ............................................................................................. 28
   8.3. SPEED OF FLIGHT ................................................................................................... 29
   8.4. REAL-TIME WORD SYNCHRONICITY ................................................................. 29
   8.5. Viable EXTENSIONS .............................................................................................. 29
   8.6. PRODUCT ASSESSMENT ....................................................................................... 29
1. Introduction

The following chapter presents the circumstances and goals of this thesis.

1.1. Background

1.1.1. RFN

The Swedish Defense Material Administration governs a missile test range together with The Swedish Space Corporation. The test range, called RFN or RobotFörsöksplats Norr, is a unique resource for governments and corporations in need of missile and UAV testing grounds, since it has a restricted over-land airspace measuring approximately $1650\, km^2$ [1]. Two examples of test flights at RFN are the unmanned space shuttle Phoenix and the air-to-air missile Meteor – see Figure 1.1 and Figure 1.2.

Figure 1.1 – Gripen firing a Meteor at the RFN test range
Source: Photograph used by permission of Gripen International
RFN has, among many other tools for aerospace testing, a long distance telemetry system.

### 1.1.2. Telemetry

It is important to clarify what is meant by the term telemetry. It is a way of sending information from one place to another in real-time, typically by the use of some radio frequency system [2]. During a test flight, such transmitted variables could be almost anything, for instance velocity, temperature or position [3].

Since modern telemetry applications transmit a wide range of parameters, giving each sensor a separate channel would not be very effective. Hence, there is need for a way to transmit them through the same channel. This can be done by using a specified frame of data – see chapter 6. The data streams sent are most often synchronous, in contrast to packet based communication.
A typical example RFN’s telemetry system in use is shown in Figure 1.3: A target drone is launched, and a fighter plane shoots a missile at it, all while telemetry is used to gather information about the test.

**Figure 1.3 – An example of a test flight using telemetry at RFN**

### 1.2. Objectives

RFN has, as of today, no suitable device for evaluating telemetry reception ability for a new customer test object, or for new testing conditions. Instead, it is necessary to make an expensive system test flight. A flight route can involve low altitude flight paths, especially for cruise missiles and UAVs. Such a device would also test and demonstrate the distribution, registration, real time presentation and data processing of telemetry signals – see Figure 1.4.

**Figure 1.4 – General project idea**
To date, no research or development on a mobile, general telemetry tester appears to have been done, although more establishments than RFN has the same need for telemetry reception testing.

The objectives were to develop a device:

- That fulfils all of both RFN’s initial demands and the objectives later derived by the project participants – see subchapter 2.2.
- That is modulated and easily extended.
- That in a suitable enclosure is durable enough to withstand an outdoor, airborne transport during all seasons of the year.
2. Specifications

Following chapter one with the main reasons for this project, this chapter describes the various requests and project specifications.

2.1. Initial requests from RFN

RFN possessed three radio transmitters and a suitable flight approved cylindrical enclosure, designated SM6 – see Figure 3.2. If these were to be used, it would be possible to fly the Janus as cargo under a helicopter, while transmitting on three different carrier frequencies, without the need of a new flight approval.

This helicopter would fly in a certain route, to simulate a free flying test flight.

The possibility to mount the device on a car, a boat or a radio tower was also requested.

2.2. Required specifications

- A sturdy design was required due to the physical treatment of the finished product. For instance, the fact that the Janus would be suspended under a helicopter implied that the electronics had to be rugged to withstand the vibrations.
- Further, the environment of different seasons called for an extended temperature range, since the Vidsel winter can present temperatures below -30° C.
- The Janus should deliver up to 20 Mbit/s synchronous bit streams including real-time data.
- The Janus has to be self-powered during flight, but should otherwise have the possibility to be driven by external power.
- There is a limit as to how wide the hardware can be, since the SM6 body has an inside radius of 82 mm.

2.3. Considering project specifications

The significance of the project specifications are vital to this thesis, and how the Janus prototype has been realized. For instance, this kind of environmental and bit rate requirements on an electronic device raises the importance of choosing proper components.
3. System description

With the project specifications given in chapter two, this chapter outlines the design of the Janus system, describing it from two different points of view.

3.1. System architecture

The requested high bit rate prohibited the use of some PC to do all the data processing. The solution was found in combining a PC104 with custom-built electronics, an electronic front-end. This way, the PC104 is relieved from constant data processing and can divert its attention to real-time sensors.

More spelled out, the Janus design can be looked upon as a chain. It begins with the PC104 CPU that reads data from the real-time sensors. It is connected to an IO card, which in turn is attached to the front-end that sends a synchronized data stream to the transmitters – see Figure 3.1.

![Figure 3.1 – A schematic showing the physical entities and how they connect](image)

The PC104 hosts a web based user interface and low level software, both described in chapter five.
The PC104 and front-end will all fitted into a PC104 enclosure. The enclosure, GPS and power pack will be mounted inside the SM6 body – see the suggested mounting in Figure 3.2.

![Diagram of Janus mounting in SM6 tow target]

Figure 3.2 – Janus mounting in SM6 tow target

Beyond the electronics, the SM6 body will be weighted down during a test flight, in order to keep it from catching wind and hit the helicopter rotor blades.
3.2. A virtual aspect

It is possible to describe the Janus system design in terms of abstraction layers, spanning from the user that connects to the user interface, to the transmitted radio signal that is received by the on-ground telemetry tracking antennas. For instance, an invocation of the Janus system could take the following form – see Figure 3.3:

1. A user connects to the HTML based user interface with a common web browser.
2. A PHP script processes the setup that the user has supplied, and puts Janus in run-time mode by calling executables written in C.
3. C programs controls all traffic to and from the IO card.
4. The IO card transmits the setup and data to and from the front-end.
5. The external hardware sends a synchronized PCM pulse train to the transmitter input.
6. The transmitter broadcasts the signal via a 3 dB divider.

![Digital abstraction layers](image)

Figure 3.3 – Digital abstraction layers

3.3. Concerning the system description

The simplifications shown in Figure 3.1 and Figure 3.3 are important, in order to get a deeper understanding of how the Janus system works. Chapter four and five deepens the system description with more information concerning the Janus hardware and software respectively.
4. **Hardware**

This chapter continues describing the system implementation, by giving a detailed description of the different hardware parts, which consists of both ordered products and custom-built electronics.

4.1. **PC104**

PC104 is an embedded computer standard that defines the bus as a version of the ISA bus, which has 104 pins, and the form factor as $90.17 \times 95.89$ mm. PC104 cards are stackable. Every stack has a CPU card and can host a number of peripheral PC104 cards [4].

The PC104 format was chosen over other industrial PC standards because the dimensions of an assembled PC104 would be suitable for the SM6 body – see Figure 3.2. Other facts that speak for this standard is that it is well-known and commonly used.

The Janus systems rugged, temperature resistant PC104 CPU card’s fundamental features are a VIA P3 660 MHz processor, 128 MB RAM, On-board Ethernet and 4 serial ports. Instead of a hard drive, it uses an extra durable industrial flash drive with 128 MB storage space.

The PC104 IO card features 48 digital IO lines with 10 kOhm pull-up resistors and dual 50-pin headers.

As a relay between the PC104 and its source of power, a DC to DC converter called HESC-104 is used. It has a wide input range of 6 to 40 V and outputs the necessary PC104 voltages of +5 V, -5 V, +12 V and -12 V. It also has 5000 W transient suppressors to clamp input power to safe levels.
The CPU card, IO card and power supply regulator card are all currently fitted into a housing of aluminium. The housing measures 5.5 inches wide, 5.75 inches high, and comes with an end cap with the standard PC connectors, including the Ethernet connector by which the web-based user interface is accessed – see subchapter 5.3.

Figure 4.1 shows the Janus PC104 stack.

![Figure 4.1 – Exploded view of the Janus PC/104 stack](image)

### 4.2. GPS

The GPS unit used in the project is a Novatel Superstar II in a rugged enclosure. The reason for this choice is that it is already at use at RFN. Its power consumption is 0.8 W and weighs 307 g.

The GPS uses a compact, active antenna with dual ceramic filters, made to sustain the outside conditions of an airplane, including corroding jet fuel and fumes.

### 4.3. FTS

FTS stands for Flight Termination System. It is a standard used not only at RFN, but at more or less every test range in the world. It is a tone based system that make it possible to terminate a test flight more or less instantly, if the test object deviates from its predetermined flight corridor.
4.4. *Electronic front-end*

When turned on, the front-end can be in one of two modes: Front-end setup or run-time.

It contains four different RAM groups, all with different purposes:

- Static RAM – Holds simple data pre-defined by the user – see subchapter 6.4.
- Dynamic RAM – Holds a customized data stream, for instance a sine wave – see subchapter 6.5.
- Real-time RAM – Holds current GPS and FTS data, updated in run-time by the PC104 – see subchapter 6.6.
- Selector RAM – Holds information about which RAM’s output is to be used at a specific address. Its output is used as chip select signals to the other RAMs.

In the setup-mode, the front-end is being configured by the PC104. This configuration consists of the programming of the bit clock and different counters as well as the writing of RAM data – see Figure 4.2.

![Figure 4.2 – Simplified schematic showing the front-end bus layout during a setup](image-url)
The front-end’s run-time purpose is to output pre-configured data combined with real-time data provided by the PC104 – see Figure 4.3.

Since the output signal is synchronous, it cannot be interrupted when the PC104, during run-time, writes data to the real-time RAM group. With the arrangement shown in Figure 4.2 and Figure 4.3, it is possible for the PC104 to address the real-time RAM individually, while the addresses to the dynamic, selector and static RAMs are generated in the counters.

The 16-bit parallel output from the RAMs is made to a serial, NRZ-L PCM data stream by shift registers.

Other parts of the front-end are:

- The bit clock – Generates a clock signal, which is used throughout the front-end. It uses one phase of a numerically controlled oscillator, making it possible to set the bit clock frequency with a 32-bit value. The first schematic was made by Ingemar Nyström, RFN.
- Word counter – Counts a number of clock signals to produce a word pulse when the predetermined word length has been reached. In other words, since the word length is programmable, it is necessary to increment the address counter when all bits of a word have been sent out from a RAM.
- Address counter – Uses the word pulses to increment the address of the static, real-time and selector RAMs up to a pre-defined level – see Figure 4.2 and Figure 4.3.
- The transmitter deviation is set up by a variable voltage, which, in broad outline, is generated by a DAC.
4.5. **Transmitters**

The transmitter carrier frequency is to be set from the user interface, to a value ranging from 2.200 to 2.500 GHz. The user interface uses a part of the front-end in order to latch a 16 bit value correlating to the frequency through to the transmitter.

Beyond the carrier frequency, it will be possible to set the transmitter power and frequency deviation.

4.6. **Power supply**

When the Janus is not mobile, it will have the possibility to be externally powered, and at the same time charge its batteries.

It was found suitable that RFN would design and deliver the power supply unit, since they have the necessary equipment and knowledge.

4.7. **Hardware conclusions**

This chapter should give knowledge of the complexity of this project. Especially the design, construction and troubleshooting of the front-end occupied a major part of the thesis timeline – see Appendix II – Project timeline.
5. **Software**

Further describing the Janus system, this chapter covers the project software, which comes down to the user interface linking to hardware control programs.

### 5.1. **Operating system**

The PC104 operating system was acquired from the PC104 supplier Diamond Systems in order to shorten development time. It is a pre-configured, boot and run-ready Linux development kit, based on the Slackware 2.4 kernel. Memory-wise, it is very compact, with only 12 MB file size and 3 MB RAM necessary for full operation.

Another benefit of this kind of operating system is that you can install a complete version on a workstation, in order to use compilers and supply additional libraries.

### 5.2. **Web server**

The Apache HTTP Server was chosen for its ease of use. It is open source, developed and maintained by an open community of developers called Apache Software Foundation [5]. It has been the most popular web server since 1996 [6].

### 5.3. **User interface**

When accessing the Janus web server, the user is introduced to a form – see Appendix I – Screenshot of user interface. The HTML form contains standard form contents such as text fields, drop-down menus, check boxes and radio buttons, in order to let the user set up the Janus system.

JavaScripts make it is possible to load and save a form state including all information from the form fields. PHP functions make it possible to ensure that the user has not entered incorrect settings in the form.

PHP is *server-side* application software, which means that the server executes the script instead of the user workstation [7]. Most of the user interface and the underlying functions such as the form parser are written in PHP.

The use of a web-based user interface has some advantages. For instance, the software a user needs, to be able to work with the interface, reduces to a standard web browser. In addition, an HTML form works essentially the same way on different operating systems.
One issue with a web based user interface could be the need of a secure connection or maybe a way to encrypt data. With some effort, this is possible, although security does not pose a problem in this particular case, since the user interface will not become public on the internet at any time.

The interface has been successfully tested on Windows, UNIX and Linux and on a variety of browsers.

5.3.1. The user interface revisited

Figure 5.1 show a detailed view of the different functions of the user interface. To the far left is the HTML form. There is also a form parser that detects which button that has been pressed and calls the right function.

When [Check setup] is pressed, a sequence of events occurs. First, the form data is checked for irregularities. Then the dynamic data is constructed and written to a temporary file on the PC104 flash disk, before the GPS is checked.

Apart from the intuitive buttons [Save setup] and [Load Setup], there is a button called [Load front-end]. When it is pressed, the execution of the hardware control programs starts – see subchapter 5.4.

If the form or some other program during the setup phase detects that the run-time process is already running, the form window will show a stop button. This procedure prevents two run-time processes to be started.
5.4. **Hardware control programs**

PHP lacks simple low level programming such as the, for this thesis, needed features of hardware communication. Instead, programs were written in C that reads GPS data from the serial port, converts a small number of analogue FTS values, writes to the ISA bus and writes data and control signals to the IO card.

The IO card resides on the ISA bus and is controlled by changing the registry data to which it is linked. The IO card makes the appropriate change on its output pins as soon as it notices that the registry value has changed.
6. The flexibility of the Janus system

This chapter describes the different ways a user can set up the Janus system to simulate the telemetry data transmitted by a real test flight object.

6.1. Telemetry frame introduction

Since the data transmitted from the Janus is synchronous, a receiver must somehow synchronize to the bit rate and every row of the sent data matrix – see table 6.1.

In the telemetry field, a whole matrix of telemetry data words is called a major frame, while minor frame means a row in that matrix.

A Janus major frame can hold up to 256 minor frames, and its minor frames can hold up to 1024 words. Table 6.1 shows an example of a data frame constructed with the Janus system, with a major frame length of 1296 bits, and with all possible contents activated.

<table>
<thead>
<tr>
<th>SFID</th>
<th>Frame-sync:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td>Minf 1:</td>
<td>1</td>
</tr>
<tr>
<td>Minf 2:</td>
<td>2</td>
</tr>
<tr>
<td>Minf 3:</td>
<td>3</td>
</tr>
<tr>
<td>Minf 4:</td>
<td>4</td>
</tr>
<tr>
<td>Minf 5:</td>
<td>5</td>
</tr>
<tr>
<td>Minf 6:</td>
<td>6</td>
</tr>
<tr>
<td>Minf 7:</td>
<td>7</td>
</tr>
<tr>
<td>Minf 8:</td>
<td>8</td>
</tr>
<tr>
<td>Minf 9:</td>
<td>9</td>
</tr>
</tbody>
</table>

6.2. Frame synchronization pattern

The simplest frame possible consists of a single minor frame of static words, where the last word is a frame synchronization pattern.

Suppose that a single row is transmitted over and over again. By searching for the synchronization word, it is possible for the receiving end to find the frequency in which the frame repeats itself.

The Janus frame synchronization pattern can span up to 4 words in the end of each row.
When the data between each frame synchronization pattern grow in size, the probability that some data equals it increases. The frame synchronization pattern can therefore be made alternating, i.e. it is inverted each other row. This procedure lowers the probability of a false lock.

The data frame in table 6.1 uses a two word long alternating frame synchronization pattern at the end of each minor frame.

### 6.3. Subframe counter

When a major frame contains more than one minor frame, a subframe counter is used, to keep track of the minor frames. The subframe counter can be placed at every location not already taken. The first minor frame gets the value 1, the second 2 and so on.

In table 6.1, the first column of the major frame contains the subframe counter.

### 6.4. Data words

In order to further simulate different customer data frames, the Janus supports word lengths from 8 to 16 bits.

A common word is a user specified word that occurs where nothing else is specified. The common words in table 6.1 have the hexadecimal value ABC.

A special word is a word that the user defines and places at a single location in the frame. The special word in table 6.1 have the hexadecimal value F, and is located at minor frame 2, word 3.

### 6.5. Dynamic words

In a real test flight data frame, some values do not need to be sampled as fast as the rest, and can be commutated – see Figure 6.1. These slow values could be a temperature, compared to accelerometer values that need to be sampled at a much higher rate. Note that this implies that values A, B, C and D in the picture is sampled one forth as fast as values 1 through 12. In other circumstances, one would say that there is a value number 4, which is multiplexed.
To simulate this in the Janus system, a *dynamic word* can be placed somewhere in the major frame. The dynamic words are all located in the dynamic RAM. The dynamic RAM address is incremented every time the RAM is read and will over time form a sine or triangle wave. One can symbolize this with Figure 6.1, saying that the static RAM is the big cycle of numbers, while the dynamic RAM is the small cycle with letters.

The period, amplitude and offset of the waves are all set from the user interface. The period can be set to only a few, or up to 32768 values. The amplitude and offset can be altered as long as the wave does not go below zero or above the maximum value possible to represent with the current word length.

![Figure 6.1 - Data stream with commutated word](image-url)
It is possible to use a dynamic value as an indicator of telemetry reception ability, by observing a wave of this kind on the receiving end [8]. If the waveform is non-continuous, like in Figure 6.2, poor reception prevails.

The dynamic word in table 6.1 is located at row 4, word 2.

### 6.6. Real-time data

The Janus real-time data consists of two different types – GPS data and FTS data.

The GPS delivers a new message once every second – see subchapter 4.2. The GPS fix message type used is GGA under the NMEA standard – see [9]. GGA contains 11 fields from which the PC104 uses the following six:

- The coordinated universal time when the fix was taken
- The latitude
- The longitude
- A bit showing the validity of the fix
- The number of satellites being tracked
- The altitude

The PC104 unpacks the GGA message and reformats it to 48 ASCII-values. These are written to the real-time RAM when in runtime, 6 words times 8 minor frames.
The FTS’ 6 analogue signals are sampled, converted to 7 bit ASCII words like the GPS data, and written to the front-end real-time RAM 10 times every second. They are positioned in the minor frame following the last GPS value.

Thus, if real-time values are to be used in a major frame it must consist of at least 9 minor frames – see columns 4 to 9 in table 6.1.

### 6.7. Other simulation features of the Janus

There are other aspects in which the Janus simulates a real test object. Most important of these are the variable bit rate, whose maximum value was one of the initial requests – see subchapter 2.2.

Other Janus features include a choice of data polarity. Inverted polarity means the output data is inverted.

Further, each word can be sent out with either its most significant bit first, or the other way around: with its least significant bit first. This is called *data alignment*.

### 6.8. Flexibility conclusions

This chapter has pointed to the very core of the Janus project so far – the tools implemented to let it transmit a simulation of a real test flight transmission.
7. Construction

With the last four chapters giving a detailed view of the project design and implementation, described below are the various methods used to manufacture the Janus prototype.

7.1. Wired test cards

It would not have been a good choice to design a circuit card following the completion of the electronic schematics, since there would have been many different versions. Instead, a wired test frame was put together.

With a wired electronic circuit is suitable since it is easy to rip up a connection or making more extensive changes.

The wired test cards were altered until their operation became approachable – see Figure 7.1. Note that this was a big part of the project, going back and forth between constructing and redesigning – see Appendix II – Project timeline.

Figure 7.1 – Wired test cards
Source: Photograph by the author

7.2. Circuit cards

The circuit card layout has been deferred until the project is in RFN’s care.
7.3.  Construction equipment and software

- Slackware Linux 10.2
  Since the PC104 Slackware system does not have a C compiler, a complete Slackware was installed on a workstation. This made it possible to compile C and simply transfer the compiled executable and necessary libraries to the PC104.

- PC104 Slackware version 2.4
  The PC104 was bought with a stripped Slackware Linux system installed on a flash disk.

- OrCAD Capture version 10.5
  All schematics of electronics were made using this program.

- Miscellaneous wiring tools
8. Evaluation

This chapter points to strengths and weaknesses of the general solution and the wired prototype, whose construction is covered in chapter seven.

8.1. Wired prototype

The limitations of the wired test frame became obvious when testing the bit clock. The bit clock wave generator is fed with a 50 MHz oscillator signal which becomes clearly visible on the output of the wave generator. In addition to, or because of this, the wired bit clock did not meet its required frequency specifications.

It was found likely that although the wired bit clock could probably have been improved with filters, the problem is likely to disappear when constructing the bit clock as a circuit card, with surface mounted circuits and a proper ground plane.

During the functionality tests of the other wired test circuits, the bit clock was generated using a function generator.

In this prototype, the Janus system has been prepared for the FTS by doing AD-conversions and by writing those temporary AD-values to the major frame during run-time.

8.1.1. Tested prototype parts

The following parts works more or less perfect, all the way from the user interface to the output signal – see Figure 3.3:

- Data alignment.
- Data polarity.
- Word, minor frame and major frame lengths.
- Frame synchronization.
- Subframe synchronization.
- Static data.
- Dynamic data.
- GPS data.

The following parts works to some extent:

- Bit clock.
- FTS data.
8.1.2. Omitted prototype parts

Due to the long development time needed, the original thesis goals were revised to an earlier point in the Janus development timeline. In the first project specification, the Janus project would be taken all the way to a flying test version.

The following parts have been deferred to when the project is under RFN’s management:

- The completion of the bit clock.
- The final FTS integration.
- The setup of the transmitters.
- The use of different data formats.
- The power supply
- The circuit card layout.
- The final mechanical construction.

8.1.3. Test equipment

- ADS100
  Advanced Decommutation System: Borrowed from RFN to check the outgoing synchronous signal. The ADS is a real-time system able to synchronize to, and decode telemetry data streams.
- Oscilloscope
- Multimeter
- Function generator

8.2. Project progress

In the project’s current state of progress, a lot of development is done. It is possible to start the PC104, and with the web-based user interface make the Janus output a PCM NRZ-L stream that contains a telemetry data frame. This frame is built by telemetry customs [10], and can contain dynamic data as stated in subchapter 6.5 and real-time data as in subchapter 6.6.

Development-wise, a Janus system test operation should not be that far away. More precisely, if the omitted project parts listed in 8.1.2 are finished, the parts left are on the receiving end.
8.3. Speed of flight

If the Janus is set up to transmit a major frame length of for instance 7200 8 bit words in 1 Mbit/s, a dynamic word is transmitted over 100 times every second and should serve as a good reception indicator. Since the FTS is sampled in 10 Hz, the GPS is the critical part as to how fast the Janus should be carried, following that it delivers only one fix per second.

8.4. Real-time word synchronicity

At first, the plan was to make the real-time data processing synchronous too, but there is no need for that. Although a GPS fix and FTS values can be output from the front-end in a dimension of 1000 times per second, they are both updated only once per second respectively ten times per second. Even if some repetitions are corrupted by interleaving or some other disturbance, the correct values should be extractable on the receiving end, provided that the reception is good enough.

8.5. Viable extensions

Since the Janus system has been built keeping future extensions in mind, it is relatively easy to add more functions to the PC104. For instance, it would not be difficult to install one or several more digital IO cards on the PC104, since each IO card uses only one hardware address. This would give a developer an amount of IO pins to work with.

Since the PC104 is a relatively common platform, there is a wide range of extension cards using this standard, and since most of the data processing in the Janus system is done separated from the PC104, it would be possible to integrate many different peripheral components to further extend the Janus system, without the PC104 getting overly busy.

8.6. Product assessment

If RFN in the future wants to introduce the Janus as a product, the market would not contain many actors, since there are not many establishments that possess a long range telemetry system. Possible customers could involve Swedish or foreign defense forces and agencies, space corporations, aircraft developers or motor racing companies.
8.7. **Visions**

Reviewing the system architecture, it would be possible to develop a Janus front-end based on a FPGA. This could have shortened the development time.

During the development I have had a possible use of a complete Janus system in mind, which includes real-time presentation as specified in Figure 1.4: Suppose the GPS deviates 10 m from the correct position and that the Janus is carried in a velocity of 25 m/s. A real-time mapping of a Janus flight might then look like in Figure 8.1.

![Figure 8.1 – Hypothetical real-time presentation](image-url)

**Figure 8.1 – Hypothetical real-time presentation**
9. Discussion

Following the project evaluation in the last chapter, the aim of this chapter is to compare the results with the goals set up.

9.1. Achievements

Compared to when this project started, I now know that:

- It is possible to configure a stripped down Linux with a fully functional web server.
- It is possible to use a workstation with a full version of the embedded Linux to simply transfer compiled C programs and the necessary libraries to a PC104.
- It is possible to use a PC104 with a web based user interface directly linked to low-level programs.

Beyond these points that others surely have accomplished before, it has also been shown that:

- It is possible to develop a bit clock, based on a wave form generator, with a bit rate configurable with 32 bits – see acknowledgement in subchapter 4.4.
- It is possible to develop a complex electronic front-end to output a very flexible telemetry frame, without the use of a FPGA or a microprocessor.

9.2. Conclusions

Below, I list comparisons with the project specifications stated in chapter two.

- With the Janus system’s current state of development, it seems possible that all of the final hardware parts could fit inside the PC104 enclosure. Thus, the dimension criterion has been met.
- Further, it seems likely that the final hardware will withstand the environmental stress, including the winter temperatures, especially since the electronics will produce a certain amount of heat.
- With a suitable power supply unit, it should not pose a problem to mount the Janus on another vehicle than a helicopter, or stationary on a mast.
- The current wired prototype has successfully delivered an approved telemetry frame in 3.6 Mbit/s, including real-time data. A circuit card version should be able to deliver 20 Mbit/s, provided that no other hardware problem arises.
- The power consumption of the GPS is stated to 0.8 W in its data sheet, and that of the PC104 together with the electronic front-end has been measured to 12 W when idle. The power used during full operation is of course higher, but constructing a battery pack that lasts for a full test flight will not pose a big challenge for RFN.

In the end, the Janus project is well on its way to become an apt system for testing long range telemetry systems.
10. References


Glossary

DAC ....................... Digital to Analogue Converter.
Front-end ................ Here: A custom-built electronic module.
FTS ....................... Flight Termination System – A tone-dial self-destruct system.
GGA ....................... Essential GPS fix Data. It is a standardized message, containing 11 elementary GPS data, for instance the current global position and the current time.
ISA ....................... A standardized data bus with 104 pins.
NRZ-L ..................... Non-Return-to-Zero-Level: a PCM coding format.
PC104 ..................... An embedded computer standard using the ISA bus
PCM ....................... Pulse-Code Modulation
PSU ....................... Power Supply Unit.
SM6 ........................ A flight container made of aluminium.
UAV ....................... Unmanned aerial vehicle.
Wired ..................... Here: A way to construct an electrical circuit by connecting ICs by thin wires instead of soldering them to a circuit card.
## Appendix I – Screenshot of user interface

### JANUS Interface v1.91

#### File commands
- **Save setup**
  - File name: [ ]
- **Load setup**
  - Choose setup file in list: [ ]
- **Refresh file list**

#### System commands
- **Check setup**
- **Reset setup**

#### Frame setup
- **Word length:** [ ]
- **Sync pattern length:** [ ]
- **Frame sync pattern:** [ ]
- **Alternating sync:** [ ]
- **Frames/Major frame:** [ ]
- **Minor frame length:** [ ]
- **Subframe ID location:** [ ]

#### Bit sync setup

**Data code:**
1. NRZ-L
2. RNRZ-L
3. Biphasel-L

**Data polarity:**
1. Normal
2. Inverted

**Data alignment:**
1. MSB first
2. LSB first

**Data rate:** [Mbps]

#### Transmitter 1 - Setup

**Power:** [ ]
**Frequency:** [GHz]
**Deviation:** [MHz]

#### Static data

**Common word data:** [ ]
**Special word:**
- **Data:** [ ]
- **Location:** [ ]

#### Dynamic data

**Shape:**
- Sine wave
- Triangular wave

**Period:** [ ]
**Amplitude:** [ ]
**Offset:** [ ]
**Location:** [ ]

#### Real-time data

**GPS & FTS:**
- **Location:** [ ]
- **Increment:** [ ]
## Appendix II – Project timeline

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Project Starts</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PC104 ordered</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PC104 has working user</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Unofficial demonstration</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Thesis finished</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RFN takes over project</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Legend: Filled squares symbolizes activity.