Internship report
HSVA Company (Hamburg Ship Model Basin)

Design and realisation of an electronical device allowing the control of strobos light used to show the cavitation phenomenon on model ships propellers in the cavitation tunnel (HYKAT)

Company tutor:
JOHANNSEN Christian

University responsible:
DE HAAN Kees

2009/2010
INTERNERSHIP REPORT
(From 03/02/2010 to 25/02/2010)

HSVA Company (Hamburg Ship Model Basin)

Design and realization of an electronic device allowing the control of strobes light used to show the cavitation phenomenon on model ships propellers in the cavitation tunnel (HYKAT).

Company tutor : JOHANNSEN Christian
University responsible : DE HAAN Kees

For Hogeschool Zeeland University of Applied Sciences
Thanks

Before beginning my report, I would like to thank all the people of HSVA for the good atmosphere in the company, especially the people of the propellers & cavitation department for their good welcome. It was very motivating to work in a good atmosphere with serious and friendly people.

I would like to do special thanks to my tutor in the company, Mr Christian Johannsen, which has always had a good listen and a good communication with me, and my tutor in the University, Mr Kees De Haan which has helped with the communication with the company and has followed my progress for the duration if the internship.

I would like also to thanks Mr Uwe Lydorf for his help during my project, making available the material needed.

Again a special thanks to Mr Herbert Bretschneider for his help and his support.

Contacts

Company tutor:
Title: Ir
Name: Christian JOHANNSEN
Company: HSVA (Hamburgische Schiffbau-Versuchanstalt GmbH)
Address: Bramfelder Strasse 164 .
D-22305 Hamburg, Germany
Website: http://www.hsva.de/
Email: johannsen@hsva.de
Phone number: +49-40-69203-339

University tutor:
Title: Ir
Name: C.D.De Haan
For Hogeschool Zeeland, Elektrotechniek
Address: Edison Weg 4,
4382 NW Vlissingen, Netherlands
Email: cdeh@xs4all.nl
Phone number: 06 24372488

Student:
Title: Mr
Name: Denis RAMBAUD
Company: Hogeschool Zeeland, mechatronics.
Address: 18 route de Molinges
39360 Rogna, France
Email: den.rambaud@gmail.com
Phone number: +33 684092171
# Table of content

Résumé......................................................................................................................................................... 5
Introduction.................................................................................................................................................... 6
I. Presentation ............................................................................................................................................... 7
   A. The Company ....................................................................................................................................... 7
      1. General Presentation......................................................................................................................... 7
      2. Propellers & Cavitation department............................................................................................... 13
      3. The team and way of working ........................................................................................................ 14
   B. Context and assignements .................................................................................................................. 16
      1. Tests in cavitation tunnel (HYKAT)................................................................................................. 16
      2. Assignements .................................................................................................................................... 18
II. Concept phase ..................................................................................................................................... 21
   A. Brainstorming..................................................................................................................................... 21
      1. Method chosen ................................................................................................................................. 21
      2. Characteristics of PICS to use ......................................................................................................... 21
      3. Choice of a PIC microcontroller .................................................................................................... 22
   B. The electronic boards ........................................................................................................................ 23
      1. Software .......................................................................................................................................... 23
      2. Problems and solutions ................................................................................................................... 24
      3. The Schematics ............................................................................................................................... 33
   C. The program ....................................................................................................................................... 34
      1. Material and software ....................................................................................................................... 34
      2. Structure and flowchart .................................................................................................................. 35
III. Tests and Realization ........................................................................................................................... 41
   A. The electronic boards ........................................................................................................................ 41
      1. Layouts ............................................................................................................................................ 41
      2. Realization and components ............................................................................................................ 42
   B. Program Realization, tests and debug ............................................................................................... 42
      1. Tests .............................................................................................................................................. 43
      2. Program ........................................................................................................................................... 45
      3. Debug ............................................................................................................................................ 53

Technical conclusion.................................................................................................................................... 55
General Conclusion ..................................................................................................................................... 56
Bibliography – references .......................................................................................................................... 57
Annexes......................................................................................................................................................... 58
Résumé

The project for the internship in HSVA Company has been realized with different steps. First of all, I have learned about the subject and what would be the application for the device I have to concept. I have watched and taken information about tests of propellers running in cavitation tunnel (HYKAT). Then I could have a better idea to begin the concept of the electronic device which has to lead the strobes light allowing the visualization of the cavitation phenomena on the model ship propellers in having fix or slow-motion pictures of it.

I have chosen what way I could use to do this device according to the requirements, and I already thought about some programming solutions in assembler for PIC microcontroller.

Then, after the choice of all the material needed, I did the concept and realized the electronic boards needed in order to test the program. The communication between the boards and the computer has been checked and the in-circuit debug of the program has been operated. Then, the program had some modifications function of this debug.

I also learned a lot about shipbuilding field in visiting the company and learning about the work done there on model ships and for research programs.
Introduction

Student in Hogeschool Zeeland University of Applied Sciences in Mechatronics as part as my French DUETI diploma for Université Lyon 1, I realize an internship of 5 months according to the study program. Indeed, this internship allow us to concretize what we have learned about project realization in our field. I have chosen to do this internship in the shipbuilding company HSVA (Hamburg Ship Model Basin) in Hamburg, Germany.

This choice was not a random choice: I have a lot of interests in shipbuilding and transport engineering. Moreover, the subject of this internship allowed me to use my skills in electronics and data processing for an application in propeller tests for model ships.

I could work on my own project following the plan I have elaborate function of the requirements. First of all I learned more about the tricks employed in cavitation tunnel for the measurement of cavitation phenomena, then I began the brainstorming to answer the requirements of my project taking in consideration the material needed. I had to learn about PIC programming in assembler language to know what settings of the microcontroller I have to use. Then the electronics board and the program has been designed and tested later.

This report shows and explains the work I have done during the internship. It will be used by the University to evaluate me and also by the company HSVA as a support and datasheet for my work.
I. Presentation

A. The Company

1. General Presentation

“For more than nine decades, the private and independent Hamburg Ship Model Basin HSVA has been at the forefront of hydrodynamic research. HSVA has influenced and led developments of testing technology, methods, standardisation and numerical procedures to solve complex problems.

Today, HSVA is also a service and consulting company for industrial customers worldwide. Highly skilled staff is trained to interact with customers to optimize products and procedures quickly with high precision. HSVA skills are acknowledged in other areas as well such as the aircraft industry. Its leading role in national and international research programs makes HSVA a most competent partner in science and services around the field of hydrodynamics and related areas.” From HSVA website

The company is operating by different ways: they are working on applied researches in all areas related to transport system and ship technologies in open water and ice as well as they are operating experiments on model ships for international customers with the help of modern test facilities and advanced technologies.

Figure 1 – HSVA fields

HSVA is working in many fields of shipbuilding. For this reason the company is divided in different departments
CAD Office (Computer Aided Design):

![Image of design and fairing in Napa / Ship model fabrication 5 axis milling machine / CAD description used for milling machine](image)

In this office they are working on the model design and building: first on the design of hull lines according to customer specifications: low power, good propeller inflows and hydrostatic properties.

Then they offer lines fairing up to production standard using the NAPA Ship Design System. The building frames are available as tables of offset printouts or on files.

Finally the Scale Model Fabrication is operated. Based on numerically control of 5 axis milling machines, it allows a fast fabrication of the ship model or other objects.

CFD (Computational Fluid Dynamics)

CFD is the fastest developing area in marine fluid dynamics. It is typically based on fundamental laws (first principles) of mass and momentum conservation.

HSVA offers a large range of service using this method. They are operating the code development for potential free surface flow and for viscous flow computations. The model basin also applies the methods to daily design and analysis work. Being able to constantly validate numerical results on experimental data, they improve in-house codes as well as their modeling strategies for commercial programs.
The fields for the use of CFD method are various:

- Wave resistance and free surface flow
- Hull form optimization
- Wake predictions
- Propeller flow
- Submarine hydrodynamics
- Manoeuvring
- Aerodynamic flow

#### HSVA Software

HSVA has also worked to develop software:

- V-Shallo is a CFD code used during design analysis and optimization.
- PPB Propeller Analysis Tool is designed to predict the performance of a propeller behind the ship and to predict the propeller induced pressures on the hull
Resistance & Propulsion

Figure 7 test running in tank / HSVA model of cruise Vessel "Millenium" (Podded-drive system)

In this department, they are working on the design optimization for hull forms and appendages. Accurate predictions of the speed/power requirement of a ship and propeller design are done. They are using the CAD systems we have seen before. They are also using modern electronic data acquisition and processing system displaying test results after each run and a fully automatic 3D wake measuring and processing system allowing the inspection of the wake field immediately after the test. Special tests for Podded-Drives and Water Jets are also done.

Figure 8 – towing tank sketch

The experiments are done in a 300m towing tank for long measuring times. I could see tests running on a large observation platform, offering optimum view of models by visitors and engineers.

Figure 9 towing tank + model
Seakeeping and manoeuvring tests are also done in the large towing tank precededly seen for resistance and propulsion experiments.

- **Seakeeping**: The tank is equipped with a double flap wavemaker at one end of the tank. Regular waves and irregular long crested seas can be simulated according to any spectral shape. In addition to standard ship models an increasing number of special ship types are being investigated such as fast monohulls, foil assisted catamarans, small waterplane twin hull catamarans and surface effect ships. The test operated are self-propulsion tests in waves or motion measurements with the free running model in long crested irregular seas at certain constant propeller speeds.

- **Manoeuvring**: The tests are done using the Computerized Planar Motion Carriage (CPMC), which provides the basis for superior predictions of the manoeuvring and course keeping qualities of surface ships. The CPMC has two fundamentally different operating modes, which enable a wide range of services and research activities. In both, the captive and the tracking mode each run is completely computer controlled from model stand-still to stand-still.
Ice & Environmental technologies

Figure 11.6 Ice Breaker tests

The Ice and Environmental Technology of HSVA is providing physical and numerical modeling and analysis services to shipyards, ship owners, oil industry governmental authorities and the Federal Ministry of Education and Research (BMBF) in the field of cold regions technology, ice hydrodynamics, and environmental hydraulics in ice covered waters.

The provided services are:

- Icebreaking Ships
- Numerical Ice technology
- Structures on ice
- Arctic Engineering
- Full scale measurements
- Measuring services
- Ice mechanics

The ice facilities are the largest ice engineering laboratories in Germany. The main feature is the 78 m long, 10 m wide and 2.5 m deep Large Ice Model Basin. The world wide largest refrigerated Arctic Environmental Test Basin which is 30m long, 6 m wide and 1.2 m deep and the Ice Mechanics Laboratories with sophisticated testing devices is also operated by HSVA.

Propellers & Cavitation

This department is working on the cavitation phenomena when propellers are running. As I worked in this department, I make its description in a special chapter.

I present especially this department and his fields of work in the next part because I have integrated it to do my internship: the subject purposed to me was directly linked to cavitation measurements in HYKAT.
2. Propellers & Cavitation department

Figure 12 – Model ship + propeller installation in HYKAT

Figure 13 – Cavitation phenomenon on a propeller

The cavitation phenomenon implied compromising results:

- Damages of propeller or rudder material because of very high energy collapses of the cavities
- Ship vibrations excited by pressure variations resulting from periodically growing and collapsing cavities
- Noise radiation

Reliable model testing are necessary to avoid at the maximum this undesired effect. The department Propellers & cavitation offers a large panel of tools to work, in which the HYKAT (Large Hydrodynamics and Cavitation Tunnel) is the main one.

Figure 14 – HYKAT

The HYKAT facility is a closed circulating cavitation tunnel with horizontal top, bottom branch submerged in a trench, numerous acoustic treatment features, variable speed, variable pressure, aeration/deaeration system. The drive system is composed of 2 electric motors (each 850 kW) driving a 3.775 m diameter seven bladed impeller, stator with nine blades. The test section is 2.80 x 1.60 x 11.00 meters.
Instrumentation of the HYKAT:

- Propeller dynamometers with drive motors inside flooded models
- Pressure sensors, hydrophones in the acoustical trough under the test section floor or in the models, computerized data collection, video system inside the flooded models, Laser-Doppler-Velocimetry (LDV)
- Planar Motion mechanism (PMM)
- High speed video system
- Particle-Image-Velocimetry (PIV)

The tests performed in the HYKAT are various:

Propeller and rudder cavitation tests, cavitation inception investigations, force measurements, determination of hydrodynamic coefficients from maneouvring tests, flow visualization, noise tests on complete hull - appendage - propulsor configurations, investigations on surface ships, submarines, torpedoes and full scale propulsor units, flow noise investigations, wake measurements.

The fields the department operates are various:

- Full Scale Investigations: see next chapter
- Hydroacoustics: problems of underwater noise generation, measurements, control and prediction. HSVA maintains an acoustic laboratory equipped with up-to-date acquisition and signal processing tools and continuously combines model testing in state-of-the-art large scale facilities (cavitation tunnels) with field studies.
- High Speed Video: a good video installation allows an observation of complete cavitation dynamics, a better understanding of pressure pulse generation and a better assessment of cavitation aggressiveness. The video synchronized recording of hull pressure, is 4500 frames/s, display 180 pictures per revolution with a resolution of 256*256 pixels. It allows also a continuous recording up to 50 revolutions.

3. The team and way of working

The department is composed of a complete and active of 14 people, engineers and technicians.
Figure 15 – Team of the department « Propellers & Cavitation »

Presentation of the team:

- JOHANNSEN Christian, Head of the department
- BÖHM Harry
- BRETSCHNEIDER Herbert
- GUTSCH Martin
- JOHANNSEN Gaby
- KRÜGER Anke
- KRÜGER Ivo
- LÜCKE Thomas, Deputy Head of the department
- LUDORF Uwe
- MANDELKAU Stefan
- POHL Martin
- SCHÖÖN Johannes
- STRECKWALL Heinrich
- WIEMER Wolfgang

The place I work:

I worked in an electronic laboratory directly close to the HYKAT with already a lot of material available. I did all the work with my personal computer in which I installed all the necessary softwares. I used volt and ampere meters, programmable waveform synthesiser as a pulse generator, solder module, wires, components…
B. Context and assignments

1. Tests in cavitation tunnel (HYKAT)

Purpose of the investigations:

- To obtain data for further improvements of model test procedures concerning the prediction of cavitation phenomena, pressure impulses and noise
- To obtain data for the validation of theoretical prediction methods
- To find possible improvements in the arrangement of propeller-hull appendages
- To find solutions to improve the propeller or rudder behavior onboard ships with Problems

Test report

I had the occasion to watch a test report of a previous test in the cavitations tunnel HYKAT for a Chinese customer for one of their boats. In this test report, first was explained the conditions of the test in the HYKAT: principle, cameras (4 cameras), what was measured: thrust, torque and rotational speed, and how, how pressure pick-ups are installed... The correspondent sketches are given in annex: Schematic of the HYKAT, photos of the propeller and the pressure pick-ups, propeller datas, location of pressure pick-ups.

With the strobe lights and the cameras during the test, they can observe the cavitation on the blades of the propeller. A scale is apparent on the propeller, and they can with that locate the apparent cavitation on the blade. At the end, we have a video of the slow motion of back and the face of the propeller with the observation of the cavitation at different angle. Prints of the video are given. Sketches are done for the back and the face at different angles. A special scale is used to recognize the different type of cavitation.
For the pressure, the results are given in a tab which is transcribed into graphs. All the harmonic amplitudes are given via a bar chart, another chart show the spatial amplitude distribution…

A conclusion is done about the efficiency and the quality of the propeller concept tested.

**Observation of a test running in the HYKAT**

I explain in this part the functionment of the test to observe the cavitation on a model propeller running in the HYKAT.

First they have to put the model in the tunnel. The model is fixed to a structure on the top of the tunnel. The structure is moving horizontally from the right to the left of the tunnel until the good position. Then the model ship is fixed in the tunnel, and the propeller installed.

All the necessary tools for the measurements are set: video camera with the right angle: most of the time, the cameras are watching the back and face of the propeller.

Strobes light are set behind the glass of the tunnel directly in front of the propeller (view from the left side, also with a horizontally direction. Those strobes lights are composed of the lights (handfassung 4035) and a module called Drelloscop 1018 from DRELLO, module taking in input the pulse sent by a computer, taking itself in input the pulses from the shaft of the propeller sent with a constant period and depending on the speed of the propeller. The strobes light are sending a blink for each revolution of the propeller, at the same position than before: like that we can have a fixed picture of the propeller letting appears the cavitation phenomena very clearly to study it.

The pulses from the shaft are sent with a constant period (the most often 360 pulses/revolution) but with a random starting point: a calibration is necessary to know on what position are the blades of the propeller. With a slow-motion view of the propeller running, we can put the blade at the position we want. For the calibration, we need to put the blade at 90, 180 or 270 ° depending on the position of the camera.
Then once that is done, and all the parameters of the propeller given (sens of rotation, number of blades), the test can occur. The user can type what blade he wants to see (blade number 1 is the one we calibrate, the position of the other is deducted) and on what angle.

The functionment of the test is directly linked to my project and is part of the assignements whose was given to me.

2. Assignements

The exact assignement is the following sentence: design and build an autonomous electronic control system allowing control the stroboscopic lights in order to measure the cavitation which operates during the propeller running, function of different parameters I will present later.

This device will allow having a slow-motion picture of the propeller, or a fix picture. For the fixed picture, the user can choose what blade of the propeller he wants to see and on what position.

This device would be used in case the computer normally leading the strobes lights shuts down. Another old similar device was build, but doesn't work properly anymore.

The device will have to follow some requirements:

- Taking in consideration the number of blades of the propeller
- Choose if we want a slow-motion picture or a fixed picture of the propeller with a push button
- For a motion picture, choose the sens (clockwise or anti-clockwise) of the motion with a button.
- For a fixed picture, choose on what blade we are working with a display
- Choose what position of this blade we want to see (every 10 degrees from 10 to 360°) with 36 buttons.
- For each button, a Led will indicate to the user what position is currently chosen: 36 Leds
- 3 calibration buttons will allow saying at the beginning that the chosen blade is at -90°, 90° or 180°.
- Indicate if the shaft of the propeller we are working on send 360 pulses/revolution (1 pulse/degree) or 1 pulse/revolution with a push button
- One button indicates if the propeller is turning clockwise or not
Figure 19 – Architecture of the device
With those informations, a little graphic with all the inputs and outputs has been worked out (in red the type of interface with the user, in blue what information is given and in green the signals in input and outputs of the control system):
II. Concept phase

A. Brainstorming

1. Method chosen

I had the free choice to work the way I want, with my own chosen method. I thought directly to the programmation microcontroller, because I had already some skills in programming. This way can allow a more compact device because the only component needed except the microcontroller himself would be the components needed for the interface between the device and the user (buttons, displays). With this way, all the signals needs to be digital datas (from 0 to 5V or more function of the microcontroller chosen), so the boards would be easier to build and the datas easier to treat.

Once I knew that, I did some research on internet to find what kind of controller I can use, with what programming language and what software and interface for the computer.

As I only worked on PIC microcontroller during my school time, I have chosen to work with a PIC microcontroller from Microchip.

I thought at the beginning to program it in C, because it’s seemed easier that assembler language. I began to do a flow chart graphic, with already some answers for C programming function depending on the assignments. But later I have chosen to use assembler because I had already used it before and because I found some very good lessons for PIC programming and they use in it the assembler language. Moreover I could see some advices telling that assembler is maybe harder, but more clear about the real way of work of the control system (it’s a source language, for example C language is translated in assembler by the programming software before to be sent in the microcontroller).

Those lessons (in French language) are very famous for their obviousness and the plenitude of information given. The internet name of the man which has done it is “Bigonoff”, the link of the website is given in annex. I have to mention him because I used a lot those lessons, and I spent a lot of time to read it before to really begin my project, in order to know what settings of the PIC could be used, and to win time for programming later.

2. Characteristics of PICS to use

At the beginning I thought the period of pulsation of the propeller had to be calculated, but actually the shaft of the propeller send pulses with the right period: 360 pulses/revolution or 1 pulse/revolution. So those pulses can be use as an external clock for the PIC using one of the timer. So in reading the datasheets and lessons about PICs, I found one function of the PIC which would fit to this purpose: a special function of the CCP module (Capture Compare and Pwm). The PIC has 2 CCP module, but we only need one (CCP1). We will use it in compare mode (the mode is chosen in the configuration register for CCP module) but without the use of the CCP pins: It consists in placing an order value in the registers CCPR (2 registers: low and high: 16 bits). Each time the timer1 (16 bits also) = CCPR, an interrupt is set.

To apply that to the project, the timer1 would be set with an external clock which is the pulses coming from the shaft of the propeller. It would count all the pulses (360 pulses/revolution: 1 pulse/degree), so we just have to put the order value in CCP function with what angle of the blade we want (after calibration) and an interrupt will allow to send a pulse to the strobes lights.
CCPIF is the bit which is changed if Timer1=CCPR1.

The use of the CCP module combined with the timer1 in counter mode will be the base of the programmatic, we will see this point again in the “Programming” part of the report.

3. Choice of a PIC microcontroller

Once I knew how will the program look like, the choice of a PIC had to be done. I was looking at the PIC table on the Microchip website to compare all the existing PIC.

I chose the PIC taking in consideration:

- the specifications we want: SPI, I2C, PWM, USB, RS232,…
- How many analogical input we need
- How many inputs/outputs
- Rated voltage: 5V (PIC XXF) or 3.3V (PICXXLF)
- Memory Flash, EEPROM or EPROM

I had to choose the right family for the PIC between 3:

- Base-Line 12 bits for an instruction (PIC 12, PIC 16)
- Mid-range 14 bits (PIC 16)
- High-end 16 bits (PIC 18, PIC 24…)

I could pick out some PICS after comparing the different settings:

- PIC 16F628: Frequency of the clock defined by only one resistor
- PIC 16F877: Lot of parameters
- **PIC 18F 4520/4620…:** Good performing
I chose the 18F4520, because it's a high performance microcontroller with many characteristics. In case, I have to change parameters in my project, this device is enough good to answer my requirements. Below its characteristics:

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Program Memory Type</td>
<td>Flash</td>
</tr>
<tr>
<td>Program Memory (KB)</td>
<td>32</td>
</tr>
<tr>
<td>CPU Speed (MIPS)</td>
<td>10</td>
</tr>
<tr>
<td>RAM Bytes</td>
<td>1,536</td>
</tr>
<tr>
<td>Data EEPROM (bytes)</td>
<td>256</td>
</tr>
<tr>
<td>Digital Communication Peripherals</td>
<td>1-A/E/USART, 1-MSSP(SPI/I2C)</td>
</tr>
<tr>
<td>Capture/Compare/PWM Peripherals</td>
<td>1 CCP, 1 ECCP</td>
</tr>
<tr>
<td>Timers</td>
<td>1 x 8-bit, 3 x 16-bit</td>
</tr>
<tr>
<td>ADC</td>
<td>13 ch, 10-bit</td>
</tr>
<tr>
<td>Comparators</td>
<td>2</td>
</tr>
<tr>
<td>Temperature Range (C)</td>
<td>-40 to 125</td>
</tr>
<tr>
<td>Operating Voltage Range (V)</td>
<td>2 to 5.5</td>
</tr>
<tr>
<td>Pin Count</td>
<td>40</td>
</tr>
</tbody>
</table>

So I finally use the PIC18F4520 because it's a complete PIC, with all the settings needed, and 40 PINS to allow bringing a sufficient number of inputs/outputs pins. It will be used with the programmer/debugger I3C and the programming software MPLABV8.0 (see part C. programming/components and software).

**B. The electronic boards**

1. **Software**

All the work has been done with my personal computer. I have installed the necessary softwares on it.

To realize the electronic boards, I used the Proteus suite. First, the software ISIS of the suite allow to draw the electronic schematics with a database compose of libraries containing a lot of pre-made components. There is also the possibility to design a new component.

Then the software ARES allows to design the layouts of the boards. The translation between ISIS and ARES is automatic, but we have to take care that all the components have a corresponding package before to pass from ISIS schematic to ARES. The components whose don't have any corresponding package, or not the right one, we need to design a new one before in ARES.
2. Problems and solutions

I will describe in this part for each assignment the material solution I have found, taking in consideration the specifications, the efficiency of the device, the comfort for the user and the PIC expectations function of the programming settings and the physical possibilities. This description is done in different parts classified function of the assignments given and is classified following the different boards designed for a better understanding.

The supply

The components need to be supplied with a 0/5V supply. A board schematic has to be designed to convert the current from 230V to 5V. The design has been found on a website called “sonelec” (see sources at the end of the report). After some researches, it fits perfectly for the supply of the device.

![Figure 21 – Supply 230V to 5V](image)

This supply is based on the use of the adjustable regulator LM317 for circuits consuming a few current. This component takes in input a voltage at least 3V more than the voltage we want in output, knowing that the maximum voltage in input is 37V: so our voltage in input has to be 37V>input>8V because we want 5V in output. For this reason I have a secondary voltage of the transformer of 9V. It has also a power of 25VA to avoid too much heating.

The value of the output voltage is determined by the resistors R1 and R2 following formula:

\[ V_{out} = 1.25 \times (1 + \frac{R2}{R1}) \]

For 5V in output, we have R1=220V and R2=660V (actually on the schematic it is R2+R4: 330+330). The rectifying of alternative voltage is done with a Graetz bridge rectifier with diodes D1 to D4 (simple diodes 1N4007) allowing a half-wave rectification.

The chemical capacitor C1 filters the rectified voltage.

The capacitors C3 & C4 avoid the parasites oscillation of the regulator.

The couple LED + resistor R3 (270) allow to show if the supply is currently running or not.
The user has to indicate the number of blade the current propeller has. To do that, a display is needed. I have chosen to put a 7 segments display for this number, linked with an instant push button which allows the incrementation of the number. To lead the 7 segments display, I have chosen to use the 74LS47 BCD to 7 segments decoder/driver.

The decoder decodes a 4 bits word (A,B,C,D inputs) into a word of 7 bits (QA:QG) allowing the display of the required segments to display the right digit from 0 to 9.

The resistors of 150 Ω are here in order to have the right voltage for the 7 leds of the 7 segments according to the Ohm law.

To assure the counting, I use the 4 bits binary counter 74LS90, counting from 0 to 9.

The counter start to 0 and increments after a pulse on CLKA. The master resets and master sets are not use (RO,R9) so they are put to the ground.

We can see that the 4 bits word is sent to the microcontroller. It will read and decode it and do the right action function of it. So for this board, 4 inputs of the PIC are needed :RA0, RA1, RA2, RA3 (linked to the PIC board with a connector).
For the anti-rebound, a simple RC filter is used. The filter RC is composed of the resistor R2 (470Ω) and the capacitor C1 (10uF). With those values the time constant is good enough to allow a anti-rebound.

The NAND door U2 (CD4093) acts like a NOT and inverts the signals (a NAND door because CD4093 was available in the company and not a simple NOT door).

Before, I tested to do an anti-rebound with a NE555 schematic, but it didn't worked properly, and finally I tested with only a RC filter and it worked perfectly and used less components so I finally chose this way.

It is exactly the same design as for the blade number, except that the 4 outputs sent to the PIC are not the same: RA4, RA5, RE0, RE3
36 buttons are needed to realize this part. I couldn’t use one pin of the PIC for each button so I did a matrix 6*6:

<table>
<thead>
<tr>
<th>Pins of the PIC (when=1)</th>
<th>RD0</th>
<th>RD1</th>
<th>RD2</th>
<th>RD3</th>
<th>RD4</th>
<th>RD5</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC3</td>
<td>PB1</td>
<td>PB6</td>
<td>PB11</td>
<td>PB16</td>
<td>PB21</td>
<td>PB26</td>
</tr>
<tr>
<td>RC4</td>
<td>PB2</td>
<td>PB7</td>
<td>PB12</td>
<td>PB17</td>
<td>PB22</td>
<td>PB27</td>
</tr>
<tr>
<td>RC5</td>
<td>PB3</td>
<td>PB8</td>
<td>PB13</td>
<td>PB18</td>
<td>PB23</td>
<td>PB28</td>
</tr>
<tr>
<td>RC6</td>
<td>PB4</td>
<td>PB9</td>
<td>PB14</td>
<td>PB19</td>
<td>PB24</td>
<td>PB29</td>
</tr>
<tr>
<td>RC7</td>
<td>PB5</td>
<td>PB10</td>
<td>PB15</td>
<td>PB20</td>
<td>PB25</td>
<td>PB30</td>
</tr>
<tr>
<td>RD6</td>
<td>PB31</td>
<td>PB32</td>
<td>PB33</td>
<td>PB34</td>
<td>PB35</td>
<td>PB36</td>
</tr>
</tbody>
</table>

In order the PIC considers that 1 button has been pushed, he waits 2 conditions: 1 for each PIN corresponding to the push button. In practice, we use DPST push buttons: one push sets 2 different outputs.

Below a little part of the schematics for only the first 5 buttons.

For each output is present a pull-down resistor (10k). A connector allows to linked this board and the board of the PIC.

No anti-rebound is needed, because it’s done by the program: the PIC will detect the first change of state from 0 to 5.
Indication of the position with leds

We need an indicator for each position and each button.

I worked on the led at the end of the schematics design, and I had already no more place on the PIC. So I had to find something working directly with the button.

I did at the beginning a schematic for the leds: for each button, a logical AND was done between its 2 outputs with a CD4081 (because they were available in the Company). Then the outputs of each AND was linked to a RC filter as an anti-rebound (10k, 10μF). Then each outputs was linked together to the reset of D flip flop (CD4013) after a diode 1N4148 for each (to do a logical OR. Each outputs is also linked to a D flip flop (CD4013) via the SET pin (CD4013 has 2 D-flip-flop, so 18 are needed). Indeed, the D flip flop CD4013 has a special state: when Reset and Set are both at 1, the outputs Q and /Q are both at 1. Like that the Reset is done on all the other LEDS but not one the one chosen which is set with the instant push button corresponding.

I tested this schematic on test-board and it worked. But 18*CD4013 + 36 diodes + 9 CD4081 (AND) and the other components needed would be too difficult to set on the board, and the schematic would be too full. With the 36 buttons, there is the need to find something easier with not so much components. The only way would be to use the PIC to lead the leds. So I had to do some places on the PIC. I did it in creating a new matrix for the other buttons (see later).
After searching other solutions, I have finally chosen to use shift registers with latch (74HC595) controlled by the PIC.

![Diagram of 74HC595](image)

**Figure 27 - Leds control with shift register with latch**

This schematic is only a part of the schematics (you can see it complete in annex) for the first 8 leds. The Leds are plugged in serial with resistor with the value according to the Ohm law. The 75HC595 has a Master Reset input. A pulse on it will reset all the leds. We don’t use the Output Enable (/OE) so it’s put to the ground. Then there are 2 clocks: 1 for the shifting, and one for the storage of the data (latch).

- **Calibration**

For the next buttons, another matrix has been done to win space on the PIC:

<table>
<thead>
<tr>
<th>Pins of the PIC</th>
<th>RB0</th>
<th>RB1</th>
<th>RB2</th>
<th>RB3</th>
</tr>
</thead>
<tbody>
<tr>
<td>RB3</td>
<td>Calib 180°</td>
<td>Calib -90°</td>
<td>Calib 90°</td>
<td></td>
</tr>
<tr>
<td>RB4</td>
<td>Sens of prop.rotation</td>
<td>Pulses/rev from the shaft</td>
<td>Fix or motion</td>
<td>Motion clock or anti_clock</td>
</tr>
</tbody>
</table>
We have 3 buttons for the calibration: one for position 90°, one for position -90° and one for position 180°. I just put those 3 buttons with pull up resistor (active when \( = 0 \)) and diodes 1N4148 for the voltage circulates in only one sens.

For the « fix or motion » button, a 3 position turning button is chosen, active on low state (pull-up resistors) with diodes 1N4148.

3 position: 1 for fix, 2 for motion (clockwise and anti-clockwise).

There is a free output because the PIC will deduct that if it’s not fix and not clockwise motion it will be anti-clockwise motion.

For the pulses/revolution choice, a normal push button is used and for the rotation propeller sens, it’s done a rotational button 2 position. The outputs are all connected to a connector to send it to the PIC. The supply 0/5V is coming also from this connector.
The PIC board

This is the main board where all the outputs from the other board are coming to connect the PIC. It contains a lot of connectors. Below the architecture of the PICS presenting all the pins and to what they are connected:

Figure 30 - PIC inputs/outputs
Some little circuits have been done for a good operation of the PIC:

- First for the oscillator, the little circuits is given in the datasheet for a good operation of the external quartz. Capacitors of 15pf are advised for high frequency oscillators (more than 4Mhz). Here, the external oscillator chosen is 20Mhz. I have chosen a high-speed quartz to be sure we don’t lose too much time and precision during the running of the program in the PIC.

- We can see also a switch for the /MCLR pin. It’s because I had to use this PIN as an output for the blade chosen if I wanted to have enough output on the PIC. So the sign is on /MCLR when we need to program or debug the PIC, and is on RE3 for the operation of the pic.

- The pulses coming from the shaft are about 12V so there was the need to adjust the voltage because the PIC takes 5V in input. For this reason the pulses are coming threw a voltage divider with 2 resistors with values according to the formula:

$$\frac{RA0/AN0/C1IN-}{RA1/AN1/C2IN-} \cdot \frac{RA2/AN2/C1IN+/VREF-/CVREF}{RA3/AN3/C1IN+/VREF+} \cdot \frac{RA4/T0CKI/C1OUT}{RA5/AN4/SS/HLVDIN/C2OUT} \cdot \frac{RA6/OSC2/CLKO}{RA7/OSC1/CLKI} \cdot \frac{RB0/AN12/FLT0/INT0}{RB1/AN10/INT1} \cdot \frac{RB2/AN8/INT2}{RB3/AN9/CCP2A} \cdot \frac{RB4/KBI0/AN11}{RB5/KBI1/PGM} \cdot \frac{PB0/KBI2/PGC}{PB1/KBI3/PGD} \cdot \frac{PC0/T1OS0/T13CKI}{PC1/T1OSI/CCP2B} \cdot \frac{PC2/CCP1/P1A}{PC3/SCK/SCL} \cdot \frac{PC4/SDI/SDA}{PC5/SDO} \cdot \frac{PC6/TX/CK}{PC7/RX/DT} \cdot \frac{RD0/PSP0}{RD1/PSP1} \cdot \frac{RD2/PSP2}{RD3/PSP3} \cdot \frac{RD4/PSP4}{RD5/PSP5} \cdot \frac{RD6/PSP6/PIC}{RD7/PSP7/PIC} \cdot \frac{RE0/RD/AN5}{RE1/WR/AN6} \cdot \frac{RE2/CS/AN7}{RE3/MCLR/VPP} \cdot \frac{VDD}{VSS} \cdot \frac{+5V}{U1} \cdot \frac{LM324}{RJ11} \cdot \frac{SIL-100-06}{SHAF'T PULSES}$$. 

![Figure 31 - PIC connections](image-url)
\[ U_{out} = \frac{R_2}{R_1+R_2} U_{in} \]

\[ U_{out} = \frac{300}{300+430} \times 12V = 5V \]

A normal operational amplifier set in voltage follower allows an output with low impedance.

- The connector called “RJ11” is the connector taking in input the signals coming from the in-circuit debugger/programmer mplab ICD3. A pull up resistor is needed for MCLR.

### 3. The Schematics.

All the schematics are given in annex.
C. The program

1. Material and software

I had to choose one software to type the program to put it the PIC. I have heard about a free software Ic_prog designed by a man and I wanted to use it because it's free and seemed well-designed and efficient. But then I chose to use mplab V8.0 because it comes also from Microchip so the datasheets are complete and as it is very used, I could find more information and helps about it. Moreover it is also a free software.

**MPLAB Integrated Development Environment (IDE)** is a free, integrated toolset for the development of embedded applications employing Microchip's PIC® and dsPIC® microcontrollers. MPLAB IDE runs as a 32-bit application on MS Windows®, is easy to use and includes a host of free software components for fast application development and super-charged debugging. MPLAB IDE also serves as a single, unified graphical user interface for additional Microchip and third party software and hardware development tools. Moving between tools is a snap, and upgrading from the free software simulator to hardware debug and programming tools is done in a flash because MPLAB IDE has the same user interface for all tools.

It has also a simulator **MPLAB SIM**, high speed software simulator for PIC and dsPIC devices with peripheral simulation, complex stimulus injection and register logging.

There is the need to use an interface between the software and the PIC via the PC. This interface would allow to program and debug the PIC directly in the circuit for an easier use. After exploring on internet, I could pick out some names:

- ICD U64, around 100€, but no compatible with mplab
- ICD U-40/64, also not compatible with mplab
- I finally chose to use the In-circuit programmer/debugger mplab ICD3 from Microchip which costs 130€. With having the complete Microchip Suite, I felt more secure to begin the programming.

It debugs and programs PIC® Flash microcontrollers and dsPIC® DSCs with the powerful, yet easy-to-use graphical user interface of MPLAB Integrated Development Environment (IDE). The MPLAB ICD 3 In-Circuit Debugger probe is connected to the design engineer's PC using a high-speed USB 2.0 interface and is connected to the target with a connector compatible with the MPLAB ICD 2 or MPLAB REAL ICE systems (RJ-11). MPLAB ICD 3 supports all Emulation headers.

I chose because it allows a real time debugging, high speed programming, is easy to use and has a low voltage emulation (2 to 5.5V).
2. Structure and flowchart

Below the flowcharts of the control system.

When a case “Run...” is put, it means that at this moment we go to another function (other flowchart) until a “return”, where we go back to the initial function we were running.

Here is the main function. The loop is continuous and allows the program running continuously.

The PIC settings are initialized function of what we want, and there is also the initialization of variables needed (flags) (those settings are described in the next part).

When an interruption condition appears, the interruption is running. The interruption structure is presented in the next flowchart.

---

**Interrupts part 1**

As there is 2 different interrupts, we need to know at the beginning of the function what interrupt is set: Timer1 overflow or CCP1?

If there is an overflow of timer1, we need to reinitialize the timer 1:

The timer 1 is 16-bits: it can count until: \( b'1111\ 1111\ 1111\ 1111' = h'FFFF' = d'65535'. \) On the next value \( (h'10000') \), an overflow occure. We need the timer1 count 359 times (359 pulses for 359 degrees). It means it needs to count 359 times before an overflow on the 360, so the starting value for timer1 has to be: \( 65536-360=65176 = h'FE98' \): timer1 low is initialized with \( h'98' \) and timer1 high with \( 'FE' \).

Then for the calibration, value of timer1 is forced. If calibration 180° is set, it means we want the blade nb 1 at 180° at this moment: we say the timer 1 he has already counted 180 times, so we put the
value $65176 + 180 = 65356 = \text{h}'\text{FF4C}'$ in the timer 1. For this position the direction of rotation doesn’t matter because in both sens the angle is $180^\circ$.

In the other hand, for other positions $(90^\circ, 270^\circ)$, the positions won’t be the same in the propeller turns clockwise or anti-clockwise. For example the position $90^\circ$ in clockwise will be $270^\circ$. For this reason we have 2 different value to put in timer 1 for both direction of the propeller: $65176 + 90 = 65266 = \text{h}'\text{FEF2}'$ and $65176 + 270 = 65446 = \text{h}'\text{FFA6}'$. Those value are inverted for the other position in the other side.

Figure 33 - flowchart interrupt part 1
Interrupt part 2:

If we want to see a slow-motion picture of the propeller, then the position is incremented and the CCP function is run (see later function "intCCP" flowchart, where the pulse is sent).

If we want to see a fix picture, the position will be determined by the 36 push buttons.

Then a position is determined: position 1 (10°): CCPR1=65176+10=65186 = h'FEA2', position 2 (20°): CCPR1=65176+20, etc.
Then for each position, a different value for a 36-bits word for the led control is sent: as explained before in the board part, the value of the LED set is determined by the position of the bit set in this word. After that, the LEDS control function is run, taking in account the value of the variable VarLed (see later function "LEDS").

Then the position is modified function of the number of blade and the blade chosen. If blade number 1, doesn’t matter how many blade there is, the current value is kept because this value has been calculated for blade 1.

For example, if there is 3 blades and blade 2 is chosen:

\[ \frac{360}{3}=120^\circ \] separates each blade, so blade 2 is 120° after blade 1, so we add 120 (\text{h'78'}) to CCPR1. But we don’t have to go up than the maximum value of TMR1 (FFFF) otherwise an overflow interrupt will occure and the calculation won’t be well done. So for the last 120° (>h'FF88'), we need to decrease 240 to CCPR1 and not add 120. It will be as if we add 120°. The same schematic is followed for all numbers of blades. After that, the CCP interrupt is run and finally we come back to the main program.
**intCCP:**

This function is the one which sent the pulse to the output (strobes light). The count of the revolution is also done (used in the slow-motion picture function). Once the count is to 30 (30 revolution), a flag is set and the counter is cleared.

---

**Figure 35 - Flowcharts blades + parameters**

**Figure 36 - Flowchart sent pulse**
This function sends the right signals to the shift registers controlling the 36 LEDs.

First a pulse is sent to the master reset to reset all the LEDs. Then we initialize a variable used to count the number of bits: 36. We test each bit to know if it is set or not, if yes we send a pulse to the DATA of the shift registers and then another to SH-CP, the clock of the shift register. Then the variable, containing the information of what LED has to be set, is shifted to the left to test the next bit after the loop. If the 36 bits have been scanned, we send a pulse to the storage register ST_CP before returning to the interrupt function.

This operation is described in the next chart from the 74HC595 datasheet showing the state of each signal of the shift register 74HC595 function of the other signals.

The DS input (data) will determine the period of the output signal. The signals coming in the register clock (SH_CP) is then stored (8 bits). The clock ST_CP allows to release the data stored. x pulses on ST_CP will set the Qx output.
III. Tests and Realization

A. The electronic boards

1. Layouts

With the software ARES I realized the layouts of the boards. The difficulty was to think about the organization of the interface, and also to do schematics as easy as possible. I realize the tracks with hands, or when it was too long and complicated and on 2 layers with an auto-router included in the software. I finalized than the tracks with the hands.

I had to create new packages for the buttons or the transformer for example. I have chosen to separate the functions in different boards because it’s much easier to use and to find the problems. Also it avoid that if one thing in a board doesn’t work properly, the other doesn’t work neither.

The layouts creation was a long part because of the choice of the components, be sure it’s the right size on the layout, avoid 2 layers when routing...

The most complicated board was the one with the 36 buttons and 36 leds, because the number of components is very big and because they have a defined position (circle) which is not necessarily the most convenient for an easy routing. I had to do it on 2 layers (both side of the board is printed). The 2 sides are linked with the components directly or with a “via”.

When some points were impossible to solve and I didn’t wanted to 2 a 2-layouts board, I just put some drills connected to nothing, and I just have to link the drills together with a wire once the board done.

We need also to take care about the size of the holes to do: if the chips are too small, once the holes done to put the components or the iron piece which will do the bridge in the case of the “via”, then there won’t be enough copper to conduct (the copper will go away with the drilling).

Below a little part of the layout for the board with the 36 buttons and leds. The 2 colours of tracks correspond to the 2 layers. The yellow drills are the “via”allowing to do the connections between the 2 sides. The design for the buttons (SWx) has been realized by me because there was nothing already existing in the software which was corresponding to the buttons I have chosen.
2. Realization and components

To realize the boards, I used PCB plates, and one big PCB plate double layer.

I printed the schematics on a special glossy paper I just get from magazines. Then I printed the schematic on the board in putting the paper directly on the copper of the PCB plate and I made it stuck with a hot iron during about 3 minutes. Then I waited the plate get colder, and put away the paper with hot water. Only the ink stay on the board, the circuits is printed on the PCB plate (we need to be careful that before to print the circuits, we need to put a mirror effect on I if we want the same orientation).

For each board I did more than one try: to efface the ink on the board, I just had to wash with aceton and then could use the board again.

Once the circuits is well-printed on the plate, I have to put it for a while in a special chemical mix composed of hot water and a chemical product (revealer). The copper is then dissolved in the solution. Only stay the copper which was under the ink printed: we have our boards. There is only the need then to clean the tracks with aceton to put away the ink, and maybe cut the board with a mechanical saw to have a good size.

After that, all the drills are done with a driller and holes between 0.7 and 1.2 mm, and the components are welded to the board.

Most of the components have been ordered to Farnell, Conrad and also Digi-Key. The price wasn’t up than 100€ + 130€ for the programmer/debugger ICD3. As there were 2 Conrad shops in Hamburg, I could go there if I needed something later.

B. Program Realization, tests and debug
1. **Tests**

Once all the boards finished and all the components fixed, I connected the boards together through the connectors. Then I begun to supply the circuits and see if all the boards are well connected.

There were some problems in the boards, and it took sometimes a long time to solve it. I had to redo it sometimes: it happened with the blades choice boards because I did a mistake in the schematic, or with the board containing the parameters buttons also. Sometimes it took a long time to find were the problem came from.

I had also problems with other thing but I could fix it directly on the board without having to redo the board.

I tested also the supply board and it worked well (it delivered a good 5V from the 230V). I already have tested it before on an experiment board (no welding) and I already solved the problem at this moment.

I finally arrive to have all the boards working properly, and could begin to set the testing installation.
On this picture you can see my realization and how it is connected for the tests. You can see on the top 2 connectors (red and black) linked to the main board with 2 wires: it corresponds to the pulses coming from the function generator. Then you can also see how the ICD3 is connected to the board. The supply is coming from a voltage generator with the 2 wires on the bottom right (brown and red). Below you can see a picture of the voltage generator and of the oscilloscope used to display so the signal, which is laid on the 50Mhz Programmable Waveform Synthesizer I use to create the pulses coming simulating the pulses from the shaft of the propeller. I used a voltage generator and not my own supply card even it worked because it was much more practice.

![Voltmeter, oscilloscope and function generator](image)

The function created with the programmable waveform synthesizer is the following:

- Burst mode
- Burst count = 1 (1 pulse per 1 pulse)
- Amplitude 12V peak/peak
- Offset of 4V (6V is not possible, it allows to have something more like 0/12V than -6V/6V)
- Function: square
- Time of sweep: 9uS (because the rated frequency of a propeller running is around 11kHz=30rev/s, so the period between 2 pulses will be 1/11000 = 9uS)
- Frequency of the main: 200kHz: as we are in burst mode, It defines the length of a pulse. So like that the high state is more little that the low state and it is more like in reality.
- Internal trigger chosen

Then the signal looks like that:

![Simulation of pulse sent from the shaft with function generator](image)

The signal is not exactly 0/12V, but the precision of the voltage doesn't matter for this case because the PIC can see as a high state a range of voltage from 3 to 5V.. It works anyway: the timer 1 counts the pulses.

Then the period of the pulse I can see in output is 360 times lower, that is normal because we send in output 1 pulse for 1 revolution and we have in input 360 pulses/revolution.
2. Program

Following the program and for each instruction, the explanation (in green) put as a commentary in the program. The program follow the flowcharts precedently given.

In the following part of the program are set the configuration bits and are chosen the settings of the PIC we need through the equivalent registers.

```
; CONFIGURATIONS

CONFIG OSC = INTIO07 ; Internal oscillator
CONFIG PWRT = ON ; Delay for the start of the device
CONFIG SWEN = ON ; Reset if stack null/underflow
CONFIG MZ7 = OFF ; Match-dog
CONFIG MCLEI = OFF ; No reset on MCLE option
CONFIG BOREN = OFF ; Reset on alimentation fall
CONFIG UVLO = OFF ; Single-Supply VCCS enabled or disabled
	(if enabled: high voltage to RB3 PIN to allow programming, RB5 dedicated to program)

; ASSIGNMENTS

// REGISTER INTRCON (standard interruption control)

INTRCONVAL EQU B'00011000'
REGISTER INTRCON

CCP1CONVAL EQU B'00001010' ; CCP1: Interrupt generated when CPU1 Timer1 value:
	CCP1 takes the position in degree enter by the user and the timer1 count from 0 to 360
REGISTER IPR ; Interrupt priority

IPC1VAL EQU B'00000100' ; IPR=1: High priority for CPU1 interrupts
REGISTER IPC1CON ; Timer1

T1CONVAL EQU B'00000100' ; bit 1 = 1: External clock from pin RC0/IO5/IO13CE (on the rising edge)
REGISTER PERIPHERAL INTERRUPT ENABLE REGISTER 1

PIE1VAL EQU B'00000001' ; TMR1 overflow interrupt enable
REGISTER OSCILLATOR CONTROL REGISTER

OSCONVAL EQU B'01100000' ; bit 6-4 IDCF=0:10: Internal Oscillator frequency select bits
	11 = 8 MHz (INTOSC drives clock directly)
REGISTER A/D CONTROL REGISTER

ADC1VAL EQU B'00001111' ; All "An" inputs as digital inputs
REGISTER ADC1 และ ADC2 ของ PIC

; Program assignments

; calculation to the 16 bits value of TIMER1: it counts 360 times
```

Figure 43 - Set bits config + assignements registers and timer1

But it is only assignments, the registers are really set later, but with those value we just have to replace with the name we have indicated. For example the word CCP1CONVAL will replace the value B'00001010'. The value for the timer1 initialization is also calculated as I explained previously.
Here the RAM is cleared. The RAM needs to be clear because when the PIC starts, random values are allocated to her. The RAM is located in the bank0 (see Data Memory Map in annex) which begin at the address 0x00. The pointer FSR is pointing on this address. Then we clear the contain of the address pointed by FSR with the instruction, and just after that we point the next address. All that is done by the instruction “clrf POSTINC0”.

Then all the ports and variables are cleared, and a value of CCP is given: when the device start, the position will be always the same.
Here all the registers are set, the timer 1 is filled and the interrupts are enable before to begin the program with the main (‘bra start’)

Here is the main program, the loop never end and can be interrupt by the 2 interrupt enabled (timer1 overflow and CCP interrupt)
Here the interrupt routine begins. The interrupt address begins at 0x008. The registers are saved in case the interrupt change them, they will be returned as they were.

Then function of what interrupt is on, we treat it.

Function on what the number of blades the propeller has, a flag is set or not. All the flags are clear at the beginning. Here on the example it’s only for 2 values of the blade number (2 and 3). For the other number it is the same architecture, and also for the blade chosen and the parameters: flags are set or not function of the button’s position.

```assembly
;ROUTINE INTERRUPTION
;
;---------------------------------------------------------------------
; CRC 0x008 ; Interruption address
; Save registers
; ---------------------------------------------------------------------
; wreg_temp ; save W
; movf STATUS, status_temp ; save STATUS
; movf BSR, bsr_temp ; BSR_TM1P located anywhere

; Figure 47 - Interrupt routine init + tmr1 overflow
btfsc PIR1, TM1IF
bra number_blades
btfsc PIR1, TM1IF
bra number_blades
;if interrupt on TM1 overflow enable and set, then treat it,
;otherwise it is an interrupt CCP, treat it

clrf TM1L
movlw HIGH Timer1VALUE
movwf TM1H
movlw LOW Timer1VALUE
movwf TM1L
; timer 1 reinitialize because overflow

clrf PIR1, TM1IF
; timer1 overflow interrupt finish
bra restorereg
; restore registers and go back in the main

; Figure 48 - Blade number + flag

number_blades
clrf flagblades
btfsc PORTA, 0
bra xxx
btfsc PORTA, 1
bra xxx
btfsc PORTA, 2
bra xxx
btfsc PORTA, 3
bra xxx
;if number of blades = 2
baf flagblades, 0

xxx
btfsc PORTA, 0
bra xxx
btfsc PORTA, 1
bra xxx
btfsc PORTA, 2
bra xxx
btfsc PORTA, 3
bra xxx
;if number of blades = 3
baf flagblades, 1
```
Here the Timer1 is forced to a value if a calibration button is pushed; taking in consideration the direction of the propeller (the button condition is represented by the bit 0 of the flag register which is set or not previously in the program function on the position on the button).

The exemple taken is only for one position (270° <-> -90°).

The other calibrations functions follow the same architecture (see flowcharts).

```
calibrations90
  btfsc PORTB, 1
  br calibrations90
  btfsc PORTB, 3
  br calibrations90
  btfsc flag, 0       ;propeller turning anti_clockwise?
  br calibrations90_clockwise
  movlw 0xFF
  movwf TMR1H
  movlw 0xFF
  movwf TMR1L
  movlw 0xFF
  movwf HIGH position
  movlw 0xFF
  movwf LOW position
  recall intposition
  br fixrotation

calibrations90_clockwise
  movlw 0x7F
  movwf TMR1H
  movlw 0xA6
  movwf TMR1L
  movlw 0xFF
  movwf HIGH position
  movlw 0xA6
  movwf LOW position
  recall intposition
  br fixrotation
```

Figure 49 - calibration operation
In this part the slow-motion is treated: if slow-motion desired, we increment or decrement (depending on what direction we want) the value of the CCP1 register of 1° all the 30 revolutions. For some special position (when the high value changes from FE to FF or from FF to FE for anti-clockwise) we need to say the next value otherwise only the low value is increment and the high value stays the same.
This function fills CCPR1 with the right value function of the position chosen. Here the number of blades is not yet taken in account because I didn’t write this part yet. I will try already without and then I will complete the program. On this example, only the first position is treated (10°). A variable is filled and sent to the led function: it defines what LED will be on if the button position 10° is pushed. Here it’s the LED 1 (bit0=1). I also did the led function only for 8 leds for the moment (8-bits). I am now writing the next part of the function to control all the leds.

```
;Interrupts for give the position function of what button push

int

btfsc  PORTC, 0 ;Test if RDI=1
brc  int0 ;No, next test
btfsc  PORTC, 3 ;Test if RDI=1
brc  int2 ;No, next test
movlw  0x42
movwf  LCD2 ;HIGH position
movlw  0x22
movwf  LCD1 ;LOW position
rcall  intposition ;Treat interruption for position 1 (10°)
btfsc  flagLED, 1
brc  return
bra  intCCP
returnn
movlw B'00000001' ; LED position 1 on
movwf  VAR
rcall  LEDS
clrf  flagLED
bcs  flagLED, 1 ;if led already ON, do nothing
bra  intCCP ;send pulse

Figure 51 - Position chosen + LED

In this part the CCP function os treated: the pulse is sent and the revolution counter incremented. The instruction “retfie FAST” allows to finish the interrupt function and come back in the main with the restoration of the registers.

```

```
;test if OCD interrupt (for security)

;******************************************************************************

btfsc  PIE1.CCP1IE ; Test if interrupt CCP1 enable
brc  rextser
btfsc  IE1.CCP1IF ; Yes, test if interrupt CCP1 running
brc  rextser
rcall  incOCD1 ; Yes, treat Interrupt CCP1
bcs  PIE1.CCP1IE ; Clear flag
brc  rextser

;restore registers
retfie  FAST ; Return with restoration of
; VAR, X, W, AS STATUS
;******************************************************************************

;******************************************************************************

bcs  PULSE_SENT ; Pulse sent to the stroboevp
incf  REV_count ; increment revolution counter
movwf  REV_count,w
bsl REV_count ; 10-REV_Count
brc  comp1
bcs  PULSE_SENT
bra  return

clrf  REV_count
bcs  REV_count
bcs  flag1

;For slow motion: change all the 30 revolutions: 1 position stay for 30 revolutions

;******************************************************************************

return
; END Interruption

;******************************************************************************

```

```

Figure 52 - Function CCP, sent pulse, revolution counter
The Led control is treated here, function of the flowchart previously given. The instruction “btfss STATUS,C” allow to control the value of the carry bit of the STATUS register: after the shift on the left of 1 bit (“rlcf Var,1”) of the word Var, one bit is “out”. The value of this bit can be controlled with the help of the “carry” bit.

The instruction “decfsz countLED,f” decrements the counter LED and put the value in “f”. The next instruction is missed if the value of “f” is equal to 0. Like that we can stop the loop when all the bits have been sent.

LEDs

```
leds
  hcf DATA_IN
  hcf SH_CP
  hcf SI_CP
  hcf RESETLEDS
  nop
  hcf RESETLEDS
  movlw 0x10
  movwf countLED ; compteur de bits = 8

loop
  rlcf Var,1
  btfsc STATUS,C
  hcf DATA_IN
  btfsc STATUS,C
  bcf DATA_IN
  hcf SH_CP ;SH_CP
  nop
  hcf SH_CP
  decfsz countLED,f ;les 8 bits sont iles envoyés? Si oui sauter instruction suivante
  brw loop ;sinon on boucle
  hcf ST_CP ;
  nop
  hcf ST_CP ;

return
```

Figure 53 - control LEDs
3. Debug

Before I realize the boards, I could debug the program using a simulator integrated in the software mplab.

There is the possibility to simulate actions on the variables with stimulus to simulate some push on the buttons or the pulses coming (clock).

For example in this window I simulate the pulses coming from the shaft on RC0. The low state of the signal will stay 100 cycles while the high state will stay 1 cycle (101 cycles for one period of the signal).

![Figure 54 - Stimulus window](image)

After solving what didn’t work well, it finally did what I expected: the timer1 was counting the pulses and an interrupt was occurred each time CCPR1=TMR1, a pulse was sent in output.

![Figure 55 - The WATCH window](image)

A little problem I took a long time to solve is for example that I couldn’t see in the “watch” window the timer1 counting (the watch window allows to see the current value of the different registers). But after a long time and many tries, I finally saw that I didn’t watch the good thing: I had to watch “TMR1_internal” and not “TMR1”, and then I could see timer1 was actually counting!

Then once the boards done and the program configured to run with mplab ICD3, I had to have a right communication between the software mplab and the PIC, and it wasn’t always good. First I had some problems with the RJ11 female plug. I took at the beginning the one which was fixed on the test board given with the ICD3, but the connection wasn’t good and it was complicated to weld properly because the chips were too small. I finally found in the company another female plug coming from a special little device allowing crossing the tracks of a RJ-11 cable. I could finally weld something good and make the connection with the board. But I had a problem I took a lot of time to solve: I could program the PIC, but not debug it. I tried to change the configuration bits, I also configured well the ports because I saw I didn’t configure them good, I have checked the current of the circuits, but I still have the problem. After some researches, I saw it could come from the external oscillator which doesn’t work, so I tried other oscillators; I tried to change the schematics... But it still didn’t work, so I finally chose to use the internal oscillator of the PIC (8MHz) and I could debug the program in the circuit.

It is just one example which makes understand how a little problem can take time to solve.
I can debug the program step by step, instruction per instruction to see what action it operates for each condition. I could also simply run the program or put a breakpoint for an instruction in the program. The program will then run until this point comes. It is really practice to find bugs, even indispensable. You can see how the interface when the debug runs in the next picture.

I am now still in the debug phase. I will do the first test in the cavitation tunnel very soon around the 10th of June.
Technical conclusion

This project has been really interesting for me because of the fields I have study for it. It has been really rewarding to work on my own. The work I have done is almost the final concept, but depending on the tests I will operate in the tunnel it might not be the final version at the end. But the tests I have already done have been concluding and I am quite confident about the reliability of the method used.

I already know that I cannot have at the end the complete product, in a box and ready to use but I think I succeeded the goals almost at 80% in doing the concept of the device which answers the assignments (without counting the work I still have to do until the end).

I think be able to say that what I have designed with a little more work on it, it will be possible to use it in the cavitation tunnel in HSVA to operate tests with a good reliability.
General Conclusion

This internship has been a really good experience for me. First, working in an international shipbuilding company was really interesting and I could see this field in detail. I could do a quality work with autonomy in a good work atmosphere. The bases I already had in electronics and programming was really put into practice, and I could learn a lot of new things with a lot of resourceful documents I read. I also learned how to organize well my work and where to find the resources I need. I knew getting rich of the experiences I had during this internship and I am happy with my almost 5 months now in the Company.

I am really interested in electronics and programming and I would like to continue my studies in this field the next year, so this internship has been a confirmation of my choice. And if I have another opportunity, I would like to get more knowledge in the field of shipbuilding and why not working in it bringing my skills in electrical engineering.

I was used to speak German and English in the company and I am very happy that now I feel more comfortable with those two languages. It was also a great challenge to work in a foreign company.
Bibliography – references


- **Wikipedia**

- **PIC tutorials and lessons**: “Le site de Bigonoff”:
  

- **Electronics forums** :
  
  Forum futura sciences :
  

  Forum abcelectronique :
  

  Forum Commentçamarche :
  
  [http://www.commentcamarche.net/forum/programmation-3](http://www.commentcamarche.net/forum/programmation-3)

A lot of **datasheets** of various components have also been consulted:

- Datasheet of DRELLOSCOP 1018 from DRELLO
- ICD User Guide
- Mplab ICD3 User Guide
- Mplab User Guide
- PIC 18F4520 (part of the datasheet in annexes)
- 74HC595, 74HC165, 74LS47, 74LS90, CD4013, CD4011, buttons, transformer, 7 segments...
Figure 57 - Main Board PIC
Figure 58 - Position + display led
Figure 59 - Blade number

Figure 60 - Blade chosen
Figure 61 – Supply 230V/5V

Figure 62 - Parameters buttons
Figure 63 - Layout Main board PIC
Figure 64 - layout position + display, double layer
Figure 65 - layout supply

Figure 66 - layout Blade number/blade chose + layout parameters buttons
PIC 18F4520:

<table>
<thead>
<tr>
<th>TABLE 1-1: DEVICE FEATURES</th>
<th>PIC 18F2420</th>
<th>PIC 18F2520</th>
<th>PIC 18F4420</th>
<th>PIC 18F4520</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operating Frequency</td>
<td>DC – 40 MHz</td>
<td>DC – 40 MHz</td>
<td>DC – 40 MHz</td>
<td>DC – 40 MHz</td>
</tr>
<tr>
<td>Program Memory (Bytes)</td>
<td>16384</td>
<td>32768</td>
<td>16384</td>
<td>32768</td>
</tr>
<tr>
<td>Program Memory (Instructions)</td>
<td>8192</td>
<td>16384</td>
<td>8192</td>
<td>16384</td>
</tr>
<tr>
<td>Data Memory (Bytes)</td>
<td>768</td>
<td>1536</td>
<td>768</td>
<td>1536</td>
</tr>
<tr>
<td>Data EEPROM Memory (Bytes)</td>
<td>256</td>
<td>256</td>
<td>256</td>
<td>256</td>
</tr>
<tr>
<td>Interrupt Sources</td>
<td>19</td>
<td>19</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>Timers</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Capture/Compare/PWM Modules</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Enhanced Capture/Compare/PWM Modules</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Serial Communications</td>
<td>MSSP, Enhanced USART</td>
<td>MSSP, Enhanced USART</td>
<td>MSSP, Enhanced USART</td>
<td>MSSP, Enhanced USART</td>
</tr>
<tr>
<td>Parallel Communications (PSP)</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>10-Bit Analog-to-Digital Module</td>
<td>10 Input Channels</td>
<td>10 Input Channels</td>
<td>13 Input Channels</td>
<td>13 Input Channels</td>
</tr>
<tr>
<td>Resets (and Delays)</td>
<td>POR, BOR, RSET Instruction, Stack Full, Stack Underflow (PWRT, OST), MCLR (optional), WDT</td>
<td>POR, BOR, RSET Instruction, Stack Full, Stack Underflow (PWRT, OST), MCLR (optional), WDT</td>
<td>POR, BOR, RSET Instruction, Stack Full, Stack Underflow (PWRT, OST), MCLR (optional), WDT</td>
<td>POR, BOR, RSET Instruction, Stack Full, Stack Underflow (PWRT, OST), MCLR (optional), WDT</td>
</tr>
<tr>
<td>Programmable High/Low-Voltage Detect</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Programmable Brown-out Reset</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Instruction Set</td>
<td>75 instructions; 83 with Extended Instruction Set Enabled</td>
<td>75 instructions; 83 with Extended Instruction Set Enabled</td>
<td>75 instructions; 83 with Extended Instruction Set Enabled</td>
<td>75 instructions; 83 with Extended Instruction Set Enabled</td>
</tr>
<tr>
<td>Packages</td>
<td>28-Pin SPDIP, 28-Pin SOIC, 28-Pin QFN</td>
<td>28-Pin SPDIP, 28-Pin SOIC, 28-Pin QFN</td>
<td>40-Pin PDIP, 44-Pin QFN</td>
<td>40-Pin PDIP, 44-Pin QFN</td>
</tr>
</tbody>
</table>
### TABLE 1.3: PIC16F4420/4520 PINOUT I/O DESCRIPTIONS

<table>
<thead>
<tr>
<th>Pin Name</th>
<th>Pin Number</th>
<th>Pin Type Buffer Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCLR</td>
<td>1</td>
<td>ST</td>
<td>Master Clear (Input) or programming voltage (Input). Master Clear (Reset) input. This pin is an active-low Reset to the device. Programming voltage input.</td>
</tr>
<tr>
<td>Vref</td>
<td>P</td>
<td>ST</td>
<td>Digital input.</td>
</tr>
<tr>
<td>RE3</td>
<td>1</td>
<td>ST</td>
<td>Digital input.</td>
</tr>
<tr>
<td>OSC1/CLKI/RA7, OSC1</td>
<td>ST</td>
<td>ST</td>
<td>Oscillator crystal or external clock input. Oscillator crystal input or external clock source input. ST buffer when configured in RC mode. analog otherwise. External clock source input. Always associated with pin function, OSC1. (See related OSC1/CLKI, OSC2/CLKO pins.)</td>
</tr>
<tr>
<td>CLKI</td>
<td>I</td>
<td>CMOS</td>
<td>Oscillator crystal output. Connects to crystal or resonator in Crystal Oscillator mode. In RC mode, OSC2 pin outputs CLKO which has 1/4 the frequency of OSC1 and denotes the instruction cycle rate. General purpose I/O pin.</td>
</tr>
<tr>
<td>RA7</td>
<td>I/O</td>
<td>TTL</td>
<td>General purpose I/O pin.</td>
</tr>
<tr>
<td>OSC2/CLKI/RA6, OSC2</td>
<td>I/O</td>
<td>—</td>
<td>Oscillator crystal or clock output.</td>
</tr>
<tr>
<td>CLKO</td>
<td>O</td>
<td>—</td>
<td>Oscillator crystal output. Connects to crystal or resonator in Crystal Oscillator mode. In RC mode, OSC2 pin outputs CLKO which has 1/4 the frequency of OSC1 and denotes the instruction cycle rate. General purpose I/O pin.</td>
</tr>
</tbody>
</table>

Legend: TTL = TTL compatible input  CMOS = CMOS compatible input or output  ST = Schmitt Trigger input with CMOS levels  I = Input  O = Output  P = Power

**Note:****
1. Default assignment for CCP2 when Configuration bit, CCP2MX, is set.
2. Alternate assignment for CCP2 when Configuration bit, CCP2MX, is cleared.

### TABLE 1.3: PIC16F4420/4520 PINOUT I/O DESCRIPTIONS (CONTINUED)

<table>
<thead>
<tr>
<th>Pin Name</th>
<th>Pin Number</th>
<th>Pin Type Buffer Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RA0/AN0</td>
<td>2</td>
<td>I/O TTL</td>
<td>PORTA is a bidirectional I/O port.</td>
</tr>
<tr>
<td>RA0</td>
<td>19</td>
<td>Analog Analog input 0.</td>
<td></td>
</tr>
<tr>
<td>RA1/AN1</td>
<td>3</td>
<td>I/O TTL</td>
<td>Digital I/O.</td>
</tr>
<tr>
<td>RA1</td>
<td>20</td>
<td>Analog Analog input 1.</td>
<td></td>
</tr>
<tr>
<td>RA2</td>
<td>4</td>
<td>I/O TTL</td>
<td>Digital I/O.</td>
</tr>
<tr>
<td>AN2</td>
<td>21</td>
<td>Analog Analog input 2.</td>
<td></td>
</tr>
<tr>
<td>AN3</td>
<td>22</td>
<td>I/O TTL</td>
<td>Comparator reference voltage input.</td>
</tr>
<tr>
<td>AN4</td>
<td>23</td>
<td>Analog Analog input 3.</td>
<td></td>
</tr>
<tr>
<td>AN5</td>
<td>24</td>
<td>I/O TTL</td>
<td>Comparator 1 output.</td>
</tr>
<tr>
<td>RA4</td>
<td>5</td>
<td>ST</td>
<td>Digital I/O.</td>
</tr>
<tr>
<td>RA3</td>
<td>22</td>
<td>Analog Analog input 1.</td>
<td></td>
</tr>
<tr>
<td>RA5</td>
<td>23</td>
<td>ST</td>
<td>Timer0 external clock input.</td>
</tr>
<tr>
<td>COUT</td>
<td>1</td>
<td>—</td>
<td>Comparator 1 output.</td>
</tr>
<tr>
<td>COUT</td>
<td>7</td>
<td>I/O TTL</td>
<td>Digital I/O.</td>
</tr>
<tr>
<td>SS</td>
<td>24</td>
<td>Analog Analog input 4.</td>
<td></td>
</tr>
<tr>
<td>SS</td>
<td>25</td>
<td>TTL</td>
<td>SPI slave select input.</td>
</tr>
<tr>
<td>SSI</td>
<td>26</td>
<td>Analog Analog input 5.</td>
<td></td>
</tr>
<tr>
<td>SSI</td>
<td>27</td>
<td>TTL</td>
<td>High/Low-Voltage Detect input.</td>
</tr>
<tr>
<td>RA7</td>
<td>1</td>
<td>—</td>
<td>Comparator 2 output.</td>
</tr>
<tr>
<td>SSI</td>
<td>28</td>
<td>Analog Analog input 6.</td>
<td></td>
</tr>
</tbody>
</table>

Legend: TTL = TTL compatible input  CMOS = CMOS compatible input or output  ST = Schmitt Trigger input with CMOS levels  I = Input  O = Output  P = Power

**Note:****
1. Default assignment for CCP2 when Configuration bit, CCP2MX, is set.
2. Alternate assignment for CCP2 when Configuration bit, CCP2MX, is cleared.
<table>
<thead>
<tr>
<th>Pin Name</th>
<th>Pin Number</th>
<th>Buffer Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RBO/INT0/FLT0/AN12</td>
<td>33</td>
<td>I/O</td>
<td>PORTB is a bidirectional I/O port. PORTB can be software programmed for internal weak pull-ups on all inputs.</td>
</tr>
<tr>
<td>RB0</td>
<td>0</td>
<td>I/O</td>
<td>Digital I/O.</td>
</tr>
<tr>
<td>NT0</td>
<td>ST</td>
<td>External Interrupt 0</td>
<td></td>
</tr>
<tr>
<td>FLT0</td>
<td>ST</td>
<td>PWM Fault input for Enhanced CCP1.</td>
<td></td>
</tr>
<tr>
<td>AN12</td>
<td>I</td>
<td>Analog input 12.</td>
<td></td>
</tr>
<tr>
<td>RBO/INT1/AN10</td>
<td>34</td>
<td>I/O</td>
<td>Digital I/O.</td>
</tr>
<tr>
<td>RB1</td>
<td>ST</td>
<td>External interrupt 1.</td>
<td></td>
</tr>
<tr>
<td>AN10</td>
<td>I</td>
<td>Analog input 10.</td>
<td></td>
</tr>
<tr>
<td>RBO/INT2/AN8</td>
<td>35</td>
<td>I/O</td>
<td>Digital I/O.</td>
</tr>
<tr>
<td>RB2</td>
<td>ST</td>
<td>External interrupt 2.</td>
<td></td>
</tr>
<tr>
<td>AN8</td>
<td>I</td>
<td>Analog input 8.</td>
<td></td>
</tr>
<tr>
<td>RB3/AN9/CCP2</td>
<td>36</td>
<td>I/O</td>
<td>Digital I/O.</td>
</tr>
<tr>
<td>AN9</td>
<td>ST</td>
<td>Capture 2 input/Compare 2 output/PWM2 output.</td>
<td></td>
</tr>
<tr>
<td>CCP2</td>
<td>ST</td>
<td>Analog input 0.</td>
<td></td>
</tr>
<tr>
<td>RB4/KB10/AN11</td>
<td>37</td>
<td>I/O</td>
<td>Digital I/O.</td>
</tr>
<tr>
<td>RB4</td>
<td>ST</td>
<td>Interrupt-on-change pin.</td>
<td></td>
</tr>
<tr>
<td>KB10</td>
<td>I</td>
<td>Analog input 11.</td>
<td></td>
</tr>
<tr>
<td>AN11</td>
<td>I</td>
<td>PWM Programming enable pin.</td>
<td></td>
</tr>
<tr>
<td>RB5/KB11/PGM</td>
<td>38</td>
<td>I/O</td>
<td>Digital I/O.</td>
</tr>
<tr>
<td>RB5</td>
<td>ST</td>
<td>Interrupt-on-change pin.</td>
<td></td>
</tr>
<tr>
<td>KB11</td>
<td>I</td>
<td>Low-Voltage ICSP™ Programming enable pin.</td>
<td></td>
</tr>
<tr>
<td>PGM</td>
<td>I</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RB6/KB12/PGC</td>
<td>39</td>
<td>I/O</td>
<td>Digital I/O.</td>
</tr>
<tr>
<td>RB6</td>
<td>ST</td>
<td>Interrupt-on-change pin.</td>
<td></td>
</tr>
<tr>
<td>KB12</td>
<td>I</td>
<td>In-Circuit Debugger and ICSP™ programming clock pin.</td>
<td></td>
</tr>
<tr>
<td>PGC</td>
<td>I</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RB7/KB13/PGD</td>
<td>40</td>
<td>I/O</td>
<td>Digital I/O.</td>
</tr>
<tr>
<td>RB7</td>
<td>ST</td>
<td>Interrupt-on-change pin.</td>
<td></td>
</tr>
<tr>
<td>KB13</td>
<td>I</td>
<td>In-Circuit Debugger and ICSP™ programming data pin.</td>
<td></td>
</tr>
<tr>
<td>PGD</td>
<td>I</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Legend:**
- TTL = TTL compatible input
- ST = Schmitt Trigger input with CMOS levels
- I = Input
- O = Output
- P = Power

**Note:** Default assignment for CCP2 when Configuration bit CCP2/MX is set.
<table>
<thead>
<tr>
<th>Pin Name</th>
<th>Pin Number</th>
<th>Buffer Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RD0/RD1/SPO</td>
<td>19 38 38</td>
<td>I/O ST</td>
<td>PORTD is a bidirectional I/O port or a Parallel Slave Port (PSf) for interfacing to a microprocessor port. These pins have TTL input buffers when PSf module is enabled.</td>
</tr>
<tr>
<td>RD0</td>
<td>19</td>
<td>I/O ST</td>
<td>Digital I/O.</td>
</tr>
<tr>
<td>PSF0</td>
<td>38</td>
<td>I/O ST</td>
<td>Parallel Slave Port data.</td>
</tr>
<tr>
<td>RD1/RD2/SPO</td>
<td>20 39 39</td>
<td>I/O ST</td>
<td>Digital I/O.</td>
</tr>
<tr>
<td>RD1</td>
<td>20</td>
<td>I/O ST</td>
<td>Parallel Slave Port data.</td>
</tr>
<tr>
<td>RD2</td>
<td>39</td>
<td>I/O TTL</td>
<td>Parallel Slave Port data.</td>
</tr>
<tr>
<td>RD3</td>
<td>41</td>
<td>I/O ST</td>
<td>Digital I/O.</td>
</tr>
<tr>
<td>RD4/RD4/SPO</td>
<td>22 41 41</td>
<td>I/O ST</td>
<td>Digital I/O.</td>
</tr>
<tr>
<td>RD5</td>
<td>42</td>
<td>I/O TTL</td>
<td>Parallel Slave Port data.</td>
</tr>
<tr>
<td>RD6/RD6/SPO</td>
<td>23 42 42</td>
<td>I/O ST</td>
<td>Digital I/O.</td>
</tr>
<tr>
<td>RD7</td>
<td>43</td>
<td>I/O TTL</td>
<td>Parallel Slave Port data.</td>
</tr>
<tr>
<td>RD8</td>
<td>23</td>
<td>I/O TTL</td>
<td>Parallel Slave Port data.</td>
</tr>
<tr>
<td>RD9</td>
<td>44</td>
<td>I/O TTL</td>
<td>Parallel Slave Port data.</td>
</tr>
<tr>
<td>RD10</td>
<td>30 5 5</td>
<td>I/O ST</td>
<td>Digital I/O.</td>
</tr>
<tr>
<td>RD11</td>
<td>31</td>
<td>I/O TTL</td>
<td>Parallel Slave Port data.</td>
</tr>
<tr>
<td>RD12</td>
<td>32</td>
<td>I/O TTL</td>
<td>Parallel Slave Port data.</td>
</tr>
<tr>
<td>RD13</td>
<td>33</td>
<td>I/O TTL</td>
<td>Enhanced CCP1 output.</td>
</tr>
<tr>
<td>RD14</td>
<td>34</td>
<td>I/O TTL</td>
<td>Enhanced CCP1 output.</td>
</tr>
<tr>
<td>RD15</td>
<td>35</td>
<td>I/O TTL</td>
<td>Enhanced CCP1 output.</td>
</tr>
</tbody>
</table>

**Legend:**
- **TTL:** TTL compatible input
- **CMOS:** CMOS compatible input or output
- **ST:** Schmitt Trigger input with CMOS levels
- **I:** Input
- **O:** Output
- **P:** Power

**Note:**
1. Default assignment for CCP2 when Configuration bit, CCP2MX, is set.
2. Alternate assignment for CCP2 when Configuration bit, CCP2MX, is cleared.

---

<table>
<thead>
<tr>
<th>Pin Name</th>
<th>Pin Number</th>
<th>Buffer Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RD0/RD1/SPO</td>
<td>8 25 25</td>
<td>I/O ST</td>
<td>PORTE is a bidirectional I/O port.</td>
</tr>
<tr>
<td>RD0</td>
<td>8</td>
<td>I/O ST</td>
<td>Digital I/O.</td>
</tr>
<tr>
<td>RD1</td>
<td>25</td>
<td>I/O ST</td>
<td>Read control for Parallel Slave Port (see also WR and CS pins).</td>
</tr>
<tr>
<td>AN5</td>
<td>25</td>
<td>I/O Analog</td>
<td>Analog input E.</td>
</tr>
<tr>
<td>RE1/RE6/RE1</td>
<td>9 26 26</td>
<td>I/O ST</td>
<td>Digital I/O.</td>
</tr>
<tr>
<td>RE1</td>
<td>9</td>
<td>I/O ST</td>
<td>Write control for Parallel Slave Port (see CS and RD pins).</td>
</tr>
<tr>
<td>AN6</td>
<td>26</td>
<td>I/O Analog</td>
<td>Analog input E.</td>
</tr>
<tr>
<td>RE2/RE7/RE2</td>
<td>10 27 27</td>
<td>I/O ST</td>
<td>Digital I/O.</td>
</tr>
<tr>
<td>RE2</td>
<td>10</td>
<td>I/O ST</td>
<td>Chip Select control for Parallel Slave Port (see related RD and WR).</td>
</tr>
<tr>
<td>AN7</td>
<td>27</td>
<td>I/O Analog</td>
<td>Analog input F.</td>
</tr>
</tbody>
</table>

**Legend:**
- **TTL:** TTL compatible input
- **CMOS:** CMOS compatible input or output
- **ST:** Schmitt Trigger input with CMOS levels
- **I:** Input
- **O:** Output
- **P:** Power

**Note:**
1. Default assignment for CCP2 when Configuration bit, CCP2MX, is set.
2. Alternate assignment for CCP2 when Configuration bit, CCP2MX, is cleared.
2.0 OSCILLATOR CONFIGURATIONS

2.1 Oscillator Types

PIC18F2420/2520/4420/4520 devices can be operated in ten different oscillator modes. The user can program the Configuration bits, FOSC<3:0>, in Configuration Register 1H to select one of these ten modes:

1. LP  Low-Power Crystal
2. XT  Crystal/Resonator
3. HS  High-Speed Crystal/Resonator
4. HSPLL High-Speed Crystal/Resonator with PLL Enabled
5. RC  External Resistor/Capacitor with Fosc/4 Output on RA8
6. RCIO External Resistor/Capacitor with I/O on RA8
7. INTIO1 Internal Oscillator with Fosc/4 Output on RA6 and I/O on RA7
8. INTIO2 Internal Oscillator with I/O on RA6 and RA7
9. EC  External Clock with Fosc/4 Output
10. ECI O External Clock with I/O on RA6

2.2 Crystal Oscillator/Ceramic Resonators

In XT, LP, HS or HSPLL Oscillator modes, a crystal or ceramic resonator is connected to the OSC1 and OSC2 pins to establish oscillation. Figure 2-1 shows the pin connections.

The oscillator design requires the use of a parallel cut crystal.

Note: Use of a series cut crystal may give a frequency out of the crystal manufacturer's specifications.

Figure 2-1: CRYSTAL/CERAMIC RESONATOR OPERATION (XT, LP, HS OR HSPLL CONFIGURATION)

Note 1: See Table 2-1 and Table 2-2 for initial values of C1 and C2.
2: A series resistor (Rs) may be required for AT strip cut crystals.
3: Rs varies with the oscillator mode chosen.

TABLE 2-1: CAPACITOR SELECTION FOR CERAMIC RESONATORS

<table>
<thead>
<tr>
<th>Mode</th>
<th>Freq</th>
<th>OSC1</th>
<th>OSC2</th>
</tr>
</thead>
<tbody>
<tr>
<td>XT</td>
<td>3.58 MHz</td>
<td>15 pF</td>
<td>15 pF</td>
</tr>
<tr>
<td></td>
<td>4.19 MHz</td>
<td>15 pF</td>
<td>15 pF</td>
</tr>
<tr>
<td></td>
<td>4 MHz</td>
<td>30 pF</td>
<td>30 pF</td>
</tr>
<tr>
<td></td>
<td>4 MHz</td>
<td>50 pF</td>
<td>50 pF</td>
</tr>
</tbody>
</table>

Capacitor values are for design guidance only. Different capacitor values may be required to produce acceptable oscillator operation. The user should test the performance of the oscillator over the expected Vcc and temperature range for the application. See the notes following Table 2-2 for additional information.

Note: When using resonators with frequencies above 3.5 MHz, the use of HS mode, rather than XT mode, is recommended. HS mode may be used at any Vcc for which the controller is rated. If HS is selected, it is possible that the gain of the oscillator will overdrive the resonator. Therefore, a series resistor should be placed between the OSC2 pin and the resonator. As a good starting point, the recommended value of Rs is 330Ω.
TABLE 2-2: CAPACITOR SELECTION FOR CRYSTAL OSCILLATOR

<table>
<thead>
<tr>
<th>Osc Type</th>
<th>Crystal Freq</th>
<th>Typical Capacitor Values Tested</th>
<th>C1</th>
<th>C2</th>
</tr>
</thead>
<tbody>
<tr>
<td>LP</td>
<td>32 kHz</td>
<td>30 pF</td>
<td>30 pF</td>
<td></td>
</tr>
<tr>
<td>XT</td>
<td>1 MHz</td>
<td>15 pF</td>
<td>15 pF</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4 MHz</td>
<td>15 pF</td>
<td>15 pF</td>
<td></td>
</tr>
<tr>
<td>HS</td>
<td>4 MHz</td>
<td>15 pF</td>
<td>15 pF</td>
<td></td>
</tr>
<tr>
<td></td>
<td>10 MHz</td>
<td>15 pF</td>
<td>15 pF</td>
<td></td>
</tr>
<tr>
<td></td>
<td>20 MHz</td>
<td>15 pF</td>
<td>15 pF</td>
<td></td>
</tr>
<tr>
<td></td>
<td>25 MHz</td>
<td>0 pF</td>
<td>5 pF</td>
<td></td>
</tr>
<tr>
<td></td>
<td>25 MHz</td>
<td>15 pF</td>
<td>15 pF</td>
<td></td>
</tr>
</tbody>
</table>

Capacitor values are for design guidance only. These capacitors were tested with the crystals listed below for basic start-up and operation. These values are not optimized.

Different capacitor values may be required to produce acceptable oscillator operation. The user should test the performance of the oscillator over the expected Vcc and temperature range for the application.

See the notes following this table for additional information.

Crystals Used:

<table>
<thead>
<tr>
<th>Frequency</th>
<th>Crystal</th>
</tr>
</thead>
<tbody>
<tr>
<td>32 kHz</td>
<td>4 MHz</td>
</tr>
<tr>
<td>25 MHz</td>
<td>10 MHz</td>
</tr>
<tr>
<td>1 MHz</td>
<td>20 MHz</td>
</tr>
</tbody>
</table>

**Note 1:** Higher capacitance increases the stability of the oscillator but also increases the start-up time.

2. When operating below 3V Vcc, or when using certain ceramic resonators at any voltage, it may be necessary to use the HS mode or switch to a crystal oscillator.

3. Since each resonator/crystal has its own characteristics, the user should consult the resonator/crystal manufacturer for appropriate values of external components.

4. Rs may be required to avoid overdriving crystals with low drive level specification.

5. Always verify oscillator performance over the Vcc and temperature range that is expected for the application.

An external clock source may also be connected to the OSC1 pin in the HS mode, as shown in Figure 2-2.

**FIGURE 2-2:** EXTERNAL CLOCK INPUT OPERATION (HS OSC CONFIGURATION)

2.3 External Clock Input

The EC and ECIO Oscillator modes require an external clock source to be connected to the OSC1 pin. There is no oscillator start-up time required after a Power-on Reset or after an exit from Sleep mode.

In the EC Oscillator mode, the oscillator frequency divided by 4 is available on the OSC2 pin. This signal may be used for test purposes or to synchronize other logic. Figure 2-3 shows the pin connections for the EC Oscillator mode.

**FIGURE 2-3:** EXTERNAL CLOCK INPUT OPERATION (EC CONFIGURATION)

The ECIO Oscillator mode functions like the EC mode, except that the OSC2 pin becomes an additional general purpose I/O pin. The I/O pin becomes bit 6 of PORTA (RA6). Figure 2-4 shows the pin connections for the ECIO Oscillator mode.

**FIGURE 2-4:** EXTERNAL CLOCK INPUT OPERATION (ECIO CONFIGURATION)

RAMBAUD Denis
FIGURE 5-6: DATA MEMORY MAP FOR PIC18F2520/4520 DEVICES

When 'a' = 0:
The BSR is ignored and the Access Bank is used.
The first 128 bytes are general purpose RAM (from Bank 0).
The second 128 bytes are Special Function Registers (from Bank 15).

When 'a' = 1:
The BSR specifies the Bank used by the instruction.

Access Bank
Access RAM Low
Access RAM High (SFRs)

FIGURE 5-7: USE OF THE BANK SELECT REGISTER (DIRECT ADDRESSING)

Note:
1: The Access RAM bit of the instruction can be used to force an override of the selected bank (BSR<3:0>) to the registers of the Access Bank.
2: The MOVFF instruction embeds the entire 12-bit address in the instruction.
# TABLE 5-2: PIC18F2420/2520/4420/4520 REGISTER FILE SUMMARY

<table>
<thead>
<tr>
<th>File Name</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
<th>Value on POR, BOR</th>
<th>Details on page</th>
</tr>
</thead>
<tbody>
<tr>
<td>TO3U</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>-0.00000</td>
<td>49, 54</td>
</tr>
<tr>
<td>TO3H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0000 0000</td>
<td>49, 54</td>
</tr>
<tr>
<td>TO3L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0000 0000</td>
<td>49, 54</td>
</tr>
<tr>
<td>STKPTR</td>
<td>STKFUL</td>
<td>STKUNF</td>
<td></td>
<td></td>
<td>SP4</td>
<td>SP3</td>
<td>SP2</td>
<td>SP1</td>
<td>00.00</td>
<td>49, 56</td>
</tr>
<tr>
<td>PCLATH</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>-0.00000</td>
<td>49, 54</td>
</tr>
<tr>
<td>PCL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0000 0000</td>
<td>49, 54</td>
</tr>
<tr>
<td>TBLPRFU</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>bit 21</td>
<td>Program Memory Table Pointer Upper Byte (TBLPTR&lt;20:16&gt;)</td>
<td>0000 0000</td>
<td>49, 76</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TBLPRHV</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Program Memory Table Pointer Upper Byte (TBLPTR&lt;15:9&gt;)</td>
<td>0000 0000</td>
<td>49, 76</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TBLPRVL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Program Memory Table Pointer Lower Byte (TBLPTR&lt;7:0&gt;)</td>
<td>0000 0000</td>
<td>49, 76</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TABLAT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Program Memory Table Latch</td>
<td>0000 0000</td>
<td>49, 76</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PRODH</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Product Register High Byte</td>
<td>xxxx xxxx</td>
<td>49, 89</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PRODL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Product Register Low Byte</td>
<td>xxxx xxxx</td>
<td>49, 89</td>
<td></td>
<td></td>
</tr>
<tr>
<td>INTO2N</td>
<td>INTO1G</td>
<td>INTO1F</td>
<td>INTO1E</td>
<td>INTO1D</td>
<td>INTO1C</td>
<td>INTO1B</td>
<td>INTO1A</td>
<td>INTO1</td>
<td>0000 0000</td>
<td>49, 93</td>
</tr>
<tr>
<td>INTO2N</td>
<td>INTO2G</td>
<td>INTO2F</td>
<td>INTO2E</td>
<td>INTO2D</td>
<td>INTO2C</td>
<td>INTO2B</td>
<td>INTO2A</td>
<td>INTO2</td>
<td>1111 1111</td>
<td>49, 94</td>
</tr>
<tr>
<td>PRENC1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>11.00 0.00</td>
<td>49, 95</td>
</tr>
<tr>
<td>PSRDS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Indirect Data Memory Address Pointer 0 Low Byte</td>
<td>0000 0000</td>
<td>49, 69</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FSRDL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Indirect Data Memory Address Pointer 0 High Byte</td>
<td>0000 0000</td>
<td>49, 69</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PSRDS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Indirect Data Memory Address Pointer 1 Low Byte</td>
<td>xxxx xxxx</td>
<td>49, 69</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BSR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Bank Select Register</td>
<td>0000 0000</td>
<td>49, 69</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PSRDS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Indirect Data Memory Address Pointer 1 High Byte</td>
<td>0000 0000</td>
<td>49, 69</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PSRDS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Indirect Data Memory Address Pointer 2 Low Byte</td>
<td>xxxx xxxx</td>
<td>49, 69</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PSRDS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Indirect Data Memory Address Pointer 2 High Byte</td>
<td>0000 0000</td>
<td>49, 69</td>
<td></td>
<td></td>
</tr>
<tr>
<td>STATUS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>N</td>
<td>OV</td>
<td>Z</td>
<td>DC</td>
<td>C</td>
</tr>
</tbody>
</table>

Legend:
- x = unknown,
- i = unchanged,
- = unimplemented,
- = value depends on condition. Shaded cells are unimplemented, read as '0'.

Note:
1. The SBOREN bit is only available when the SBOREn+1.0 Configuration bit = 1; otherwise, it is disabled and reads as '0'. See Section 4.4.2 "Brown-out Reset (BOR)".
2. The STKFUL and STKUNF bits are not implemented on 28-pin devices and are read as '0'. If implemented on 44-pin devices, individual unimplemented bits should be interpreted as '1'.
3. The PLLF bit is only available in specific oscillator configurations; otherwise, it is disabled and reads as '0'. See Section 2.6.4 "PLL in INTOSC Mode".
4. The RE3 bit is only available when Master Clear Reset is disabled (MCLE Configuration bit = 0); otherwise, RE3 reads as '0'. This bit is read-only.
5. The RA9RA8 and their associated latch and direction bits are individually configured as port pins based on various primary oscillator modes. When disabled, these bits read as '0'.

---

[Image of the text]
### TABLE 5-2: PIC18F2420/2520/4420/4520 REGISTER FILE SUMMARY (CONTINUED)

<table>
<thead>
<tr>
<th>File Name</th>
<th>BR7</th>
<th>BR6</th>
<th>BR5</th>
<th>BR4</th>
<th>BR3</th>
<th>BR2</th>
<th>BR1</th>
<th>BR0</th>
<th>Value on POR, BOR</th>
<th>Details on Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>TMR0H</td>
<td>00000000</td>
<td>50, 126</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TMR0L</td>
<td></td>
<td>00000000</td>
<td>50, 126</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TDCON</td>
<td>11111111</td>
<td>50, 126</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OSCCON</td>
<td>00100000</td>
<td>30, 50</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HVLCON</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WDTCON</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RCON</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TMR1H</td>
<td>00000000</td>
<td>50, 132</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TMR1L</td>
<td>00000000</td>
<td>50, 132</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T1CON</td>
<td>00000000</td>
<td>50, 127</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TMR2</td>
<td>00000000</td>
<td>50, 134</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PR2</td>
<td>00000000</td>
<td>50, 134</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T2CON</td>
<td>11111111</td>
<td>50, 134</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SSPTBF</td>
<td>00000000</td>
<td>50, 133</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SSΠADD</td>
<td>00000000</td>
<td>50, 170</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SSΠSTAT</td>
<td>00000000</td>
<td>50, 170</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SSΠCON1</td>
<td>00000000</td>
<td>50, 170</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SSΠCON2</td>
<td>00000000</td>
<td>50, 170</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADRESH</td>
<td>00000000</td>
<td>50, 170</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADRESL</td>
<td>00000000</td>
<td>50, 170</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADCCON</td>
<td>00000000</td>
<td>50, 170</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADCCON2</td>
<td>00000000</td>
<td>50, 170</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CCP1H</td>
<td>00000000</td>
<td>50, 170</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CCP1L</td>
<td>00000000</td>
<td>50, 170</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CCP1CON</td>
<td>00000000</td>
<td>50, 170</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CCP2H</td>
<td>00000000</td>
<td>50, 170</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CCP2L</td>
<td>00000000</td>
<td>50, 170</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CCP2CON</td>
<td>00000000</td>
<td>50, 170</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BAUDCON</td>
<td>00000000</td>
<td>50, 170</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PWMD1CON</td>
<td>00000000</td>
<td>50, 170</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PWMD2CON</td>
<td>00000000</td>
<td>50, 170</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TMR3H</td>
<td>00000000</td>
<td>50, 170</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TMR3L</td>
<td>00000000</td>
<td>50, 170</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Legend: x = unknown, u = unchanged, _ = unimplemented, = value depends on condition. Shaded cells are unimplemented, read as ‘0’.

Note:
1. In state BOREN=1.0= Configuration bits = 0, otherwise, it is disabled and reads as ‘0’. See Section 4.4 "Brown-out Reset (BOR)."
2. These registers and/or bits are not implemented on 28-pin devices and are read as ‘0’. Reset values are shown for 44/44-pin devices; individual unimplemented bits should be interprted as ‘0’.
3. The PLLEN bit is only available in specific oscillator configurations; otherwise, it is disabled and reads as ‘0’. See Section 2.8.4 "PLL in INTOSC Mode”.
4. The RE3 bit is only available when Master Clear Reset is disabled (MCRLRE Configuration bit = 0), otherwise, RE3 reads as ‘0’. This bit is read-only.
5. RAS/PAD and their associated latch and direction bits are individually configured as port pins based on various primary oscillator modes. When disabled, these bits read as ‘0’. 

Acknowledgments: This work is supported by the French National Research Agency (ANR) within the framework of the excellence research project ANR-10-ERDF-0007 - HILIA - 'High Impact LIA for the 21st Century'.
<table>
<thead>
<tr>
<th>File Name</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
<th>Value on POR, BOR</th>
<th>Details on page.</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPBRGH</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>51,206</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SPBRG</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>51,206</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RCREG</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>51,213</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TXREG</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>51,211</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TXSTA</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>51,202</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RXSTA</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>51,203</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EEXADR</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>51,74, 83</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EEXDATA</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>51,74, 83</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EECON2</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>51,74, 83</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EECON1</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>51,74, 83</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IPR2</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>52,101</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IPR1</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>52,99</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OSCTR2</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>52,100</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OSTUNE</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>52,98</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TRISB</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>52,118</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TRISA</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>52,106</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Legend: x = unknown, u = unimplemented, 0 = value depends on condition. Shaded cells are unimplemented, read as ‘0’.

1. The SBORCN bit is only available when the BOREN bit = 0; otherwise, it is disabled and reads as ‘0’. See Section 4.4.2 “Boosted-out Reset (BOR).”
2. These registers are implemented on 28-pin devices and are read as ‘0’. Reset values are shown for 40-pin devices; individual unimplemented bits should be interpreted as ‘1’.
3. The PLEN bit is only available in specific oscillator configurations; otherwise, it is disabled and reads as ‘0’. See Section 2.8.4 “PLL In INTOSS Mode.”
4. The RE3 bit is only available when Master Clear Reset is disabled (MCLR Configuration bit = 0); otherwise, RE3 reads as ‘0’. This bit is read-only.
5. RAS/RAT and their associated latch and direction bits are individually configured as port pins based on various primary oscillator modes. When disabled, these bits read as ‘0’.