
Phase-Locked Loop Integrated Circuits

PART II PRACTICAL CLASS MANUAL

CAVENDISH LABORATORY, UNIVERSITY OF CAMBRIDGE

Head of Class: A. C. Irvine (aci20@cam.ac.uk)

Revised and updated September 2017

Based on an original experiment and manuscript by J. R. A. Cleaver, 2002



Contents

Contents	ii
Overview.....	4
1. Background	4
1.1. The Experimental System.....	4
1.2. Methodology and Goals.....	4
1.3. Development and Submission of Code	5
Computer-Based Control and Measurement	6
2. Overview of Measurement Equipment with Examples	6
2.1. PicoScope	6
2.2. Arduino UNO	7
2.2.1. Input/Output Pins	7
2.2.2. Programming the UNO.....	8
2.3. LabVIEW	12
2.3.1. LabVIEW Basics	12
2.3.2. Communicating with Instruments	14
2.4. The Integrated Measurement System	16
2.4.1. The LabVIEW PicoScope VI.....	16
Background to the Experiment.....	20
3. Introduction to Phase-Locked Loops	20
3.1. The Experiment	20
4. Phase-Locked Loop System Operation.....	21
4.1. Feedback and Stability	21
4.2. Effects of Filter Circuits on Stability	21
4.3. Tracking and Capture	22
5. The CMOS Integrated Circuit Type 4046 Phase-Locked Loop.....	23
5.1. General Outline	23
5.2. Phase Comparator Type 1: EXCLUSIVE-OR Circuit	24
5.3. Phase Comparator Type 2: Edge-Triggered Circuit	24
5.4. Voltage-Controlled Oscillator.....	26
5.5. Quiescent Conditions and Signal Capture.....	27
5.6. Noise and Interference	27
6. Integrated-Circuit Layout and Operation.....	28
The Experiment	29
7. Experimental Characterisation of the Phase-Locked Loop Integrated Circuit.....	29

7.1.	Phase Comparator Characterisation - Phase-Shift Circuit	29
7.2.	Phase-Comparator Characterisation - Type 1 Comparator	30
7.3.	Phase-Comparator Characterisation - Type 2 Comparator	30
7.4.	Voltage-Controlled Oscillator Characterisation	30
7.5.	Basic Closed-Loop Operation	31
8.	Phase-Locked Loop Integrated Circuit Applications	32
8.1.	Frequency Multiplier	32
8.2.	Transient Response and Frequency Modulation	32
8.3.	Clock Regeneration	33
8.3.1.	Characterising Clock Regeneration	34
8.3.2.	Recovering and Decoding a Real Signal	35
	References	36
	Selected Further Reading	36
	Appendices	37
A1.	Experimental Apparatus	37
A1.1.	The Prototyping Board	37
A2.	The Phase-Locked Loop as a Control System	38
A2.1.	Linear, Small-Signal Analysis for Stability	38
A2.2.	Effects of Filter Circuits on Stability	39
A2.3.	Gain, Margin, Phase Margin and Acceptable System Performance	41
A2.4.	Bode Diagrams	43
A3.	Integrated-Circuit Connections	45
A3.1.	HEF4046B and 74HC4046 Phase-Locked Loops	45
A3.2.	μ A741 Operational Amplifier	46
A3.3.	LM393A Dual Comparator	47
A3.4.	HEF4015B Dual 4-Bit Static Shift Register	47
A3.5.	HEF4020B and CD4020BE 14-Stage Binary Counters	48
A3.6.	HEF4081 B Quadruple 2-Input AND Gate	48
A4.	Operational Amplifier Circuits	49
A5.	Physical Layout of the HEF4046B	50
A5.1.	Introduction	50
A5.2.	Example 1: Four-Input NOR Gate	50
A5.3.	Process Flow	51
A5.4.	Example 2: Large Channel-Width Transistor	53

Overview

1. Background

1.1. The Experimental System

This project is essentially an investigation of the phase-locked loop, a mixed digital and analogue circuit based on the 4046 family of sixteen-pin chip packages; its operation is subtle and powerful and it has many uses. In addition, much emphasis will be placed on getting a taste of data acquisition by computer, involving the logging of relatively large data sets in an automated fashion by means of custom-built and adapted programs.

The HEF4046B phase-locked loop chip forms part of a feedback circuit which allows it to 'lock on' to a digital waveform, producing its own output which has a well-controlled frequency and phase relationship to the test signal. It consists of two principal parts; a phase comparator which produces an 'error' signal which depends on the phase difference between two inputs, and a voltage-controlled oscillator which, as the name suggests, produces a digital output of frequency dependent on the voltage applied to its input. The frequency range attainable depends on external electronic components chosen by the user.

The experimenter will investigate the performances of the phase comparators and the voltage-controlled oscillator independently, before constructing a feedback circuit, in which complex feedback behaviours will be investigated.

Data acquisition will involve elements of computerised automation. The phase comparators will be studied using an Arduino UNO microcontroller, which the experimenter will program to use as a pulse generator. The high-level graphical programming language LabVIEW will be used to record data sets and to control the experiments. Note that in order not to disadvantage those without a strong computer science background, **no great amount of programming experience is necessary, nor should such experience confer a particular advantage.**

1.2. Methodology and Goals

Some things to bear in mind:

- This project has been newly adapted from a more hands-on electronics-based experiment; as such, the timescales for completing the various elements are not known as well as they once were. It is NOT expected (nor was it in the past) that all tasks should be completed, and it is FAR more important that the appropriate experimental techniques, precautions and analysis have been applied. It is reasonably important, however, to have investigated (and understood) the phase comparators and voltage-controlled oscillator (Sections 7.1 to 7.4), and to carry out a reasonably thorough investigation of the basic closed loop (Section 7.5). The example applications of Section 8 will be helpful to your final mark if your investigation does them justice, but there are no prizes for ticking the most boxes.
- Discovering the characteristics of the devices under test is the key here, but it is an excellent opportunity to make use of your laptop to acquire a large set of data; though a manually-plotted graph with half a dozen points may prove a hypothesis, a curve (or family of curves) with hundreds or thousands of data points may be more convincing PROVIDED you do not forget the scientific justification, errors and so on. This experiment allows you to explore the power of automation, and it is expected that you will make use of it.

- A physics experiment is supposed to reveal the underlying principles of the system of interest, and is quantitative as well as qualitative.
- This project is far from perfect - intentionally - and it is crucial that you deal with and record imperfections in the appropriate manner.

If there are errors or unreasonable difficulties in this project, let me know as soon as possible (aci20@cam.ac.uk).

1.3. Development and Submission of Code

As you will already know, you are expected to submit a formal write-up of the experiment for assessment. In addition, I would like to receive any significant examples of code which you may have written. **I will NOT require ANY of your code from Section 2**, which is intended as a practice and learning exercise. However, programs you create or modify for use in the experimental phase, Sections 7 and 8, should be submitted.

You will NOT be assessed on the quality or beauty of your coding, only on your ideas of what measurements to take. The science of the experimental circuit is all that matters. Submitting code will serve the purpose of helping to figure out where any problems arose, and will hopefully discourage plagiarism (remember that passing your code to a colleague might disadvantage you once marks are moderated!).

At present, the submission method has not been finalised, but make sure you save your code in a safe place and you will be instructed during the class how to deal with it.

Computer-Based Control and Measurement

2. Overview of Measurement Equipment with Examples

2.1. PicoScope

PicoScope is a staple of the IA and IB experimental classes, and you will be thoroughly familiar with it. Consequently this manual will not go into detail about it (there is a User Guide available from the 'Help' menu). The model used in this experiment is the PicoScope 2205A [1], which has two voltage input channels and an arbitrary waveform generator (AWG), which does (in broad terms) what it says it does, but knowledge of its operation is not useful here. It has a bandwidth of 25 MHz and a sample rate of 200 MS/s. In the usual way, the PicoScope should be plugged into your laptop using the standard USB type-A-to-type-B cable provided. Occasionally it needs re-connecting if the laptop can't recognise it.

The software again will be familiar to you (Figure 1). Use the PicoScope and proprietary software whenever you see fit, remembering to calibrate the scope probes by adjusting the trimmer to avoid over/undershoot, and remember also to use the 'x10' software setting, as the probe attenuates the voltage by a factor of 10.

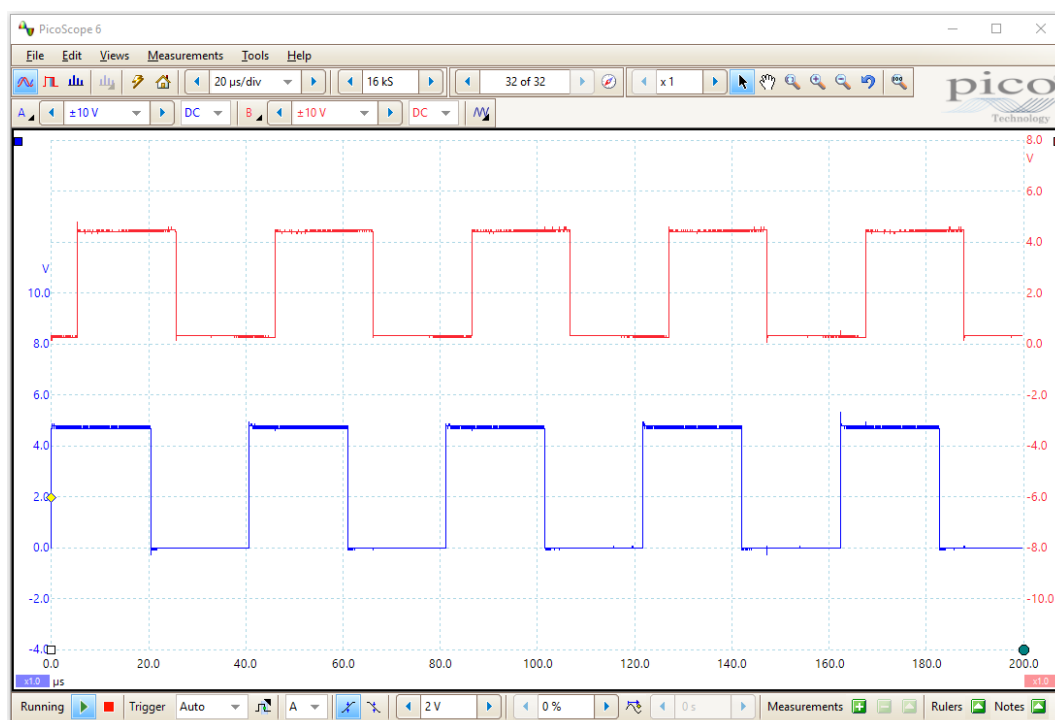


Figure 1 - PicoScope software GUI.

2.2. Arduino UNO

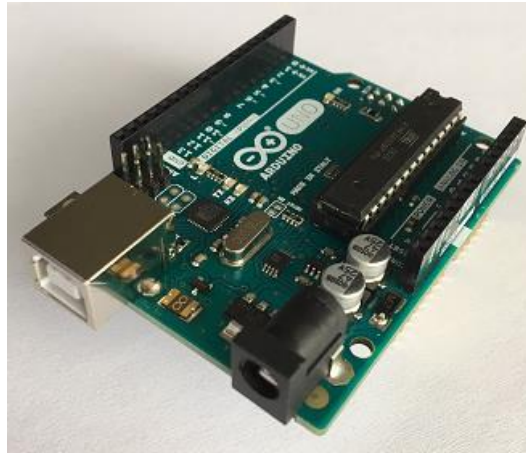


Figure 2 - Arduino UNO.

Arduino is a series of small, inexpensive microcontrollers based on open-source technology, and available in a range of specifications. It has gained a strong following amongst hobbyists, and is a powerful tool capable of sophisticated applications. It is a handy demonstrator for the principles of experimental control and measurement, whilst retaining enough drawbacks to challenge the experimenter's technique. The *UNO* model (Figure 2) will be used in this project, incorporating a 16 MHz ATmega328P processor. Power may be supplied by a transformer, but it is usually perfectly adequate (and, here, more convenient) to energise using only the UNO's A-to-B USB connector lead, through which the unit may also be programmed. For those interested, full specifications may be found online [for the Arduino UNO](#) [2] and [the ATmega328P](#) [3].

Again, remember that this section is not intended for submission within your write-up - just go through it to familiarise yourself with the Arduino.

2.2.1. Input/Output Pins

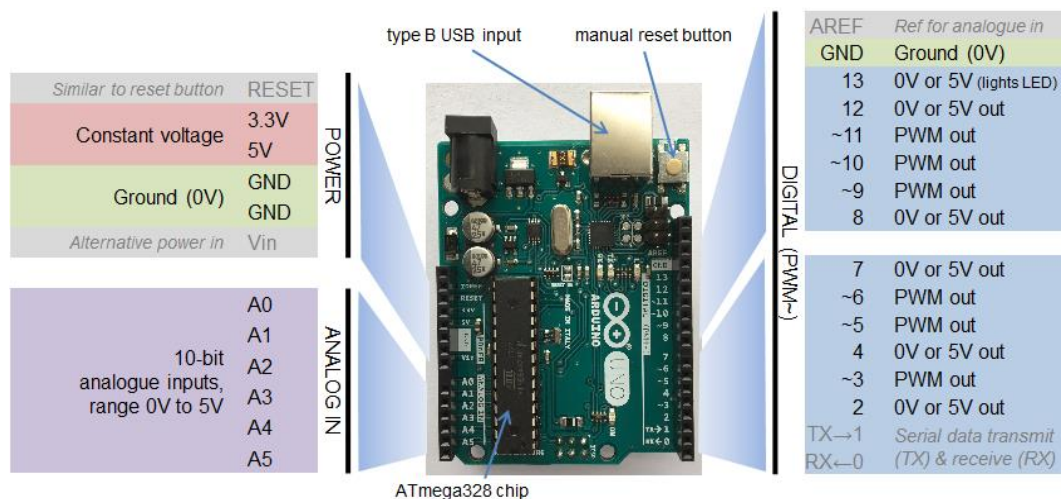


Figure 3 - Plan view of Arduino UNO board annotated with I/O pin assignments.

The UNO has two banks of input/output pins, one on each side of the board, as shown in Figure 3, as well as a manual reset button which (usefully!) can be used to restart the UNO's on-board program.

Pay no attention to those greyed out in the figure (RESET, AREF, Vin, TX→1 and RX←0), as they will play no part in this experiment. The 5 V and 3.3 V pins and the analogue inputs may well also not be needed, but you will be using a ground pin(s) as a reference level, and you will certainly require the voltage-out pins (illustrated in blue, on the right-hand side). Note that, ignoring some complexities of operation, these pins can be separated into two types. The first is a high-low or binary type, which may be made to output only 0 V or 5 V (pins 2, 4, 7, 8, 12 and 13). Of these, pin 13 is special, as it is associated with an adjacent orange light-emitting diode (LED) which indicated whether the pin is high or low. The second type of pin, numbers

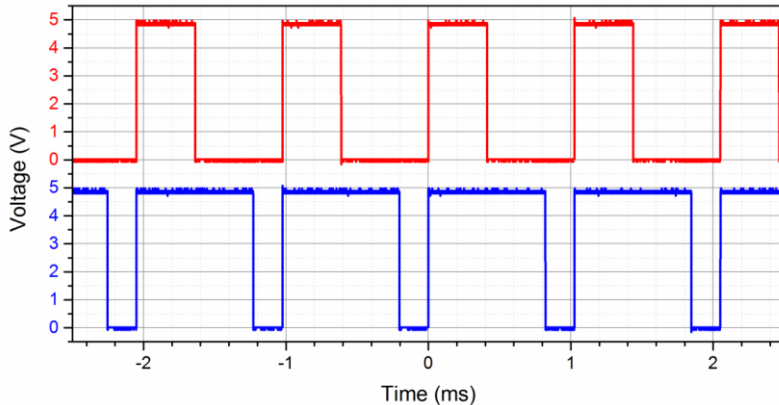


Figure 4 - Pulse-wave modulation (PWM) output voltage variation of an Arduino UNO for set mean voltages of approximately 2 V (upper) and 4 V (lower).

3, 5, 6, 9, 10 and 11, has a pulse-width modulated (PWM) output; at any given time, they must output either 0 V or 5 V, but by varying the duration of the two states when in a pulsed mode, they can be made to average an arbitrary value in between (allowing for 8-bit, or 256-state, precision). Figure 4 shows two traces in PWM mode, the upper having a mean value around 2 V, the lower 4 V. The frequency of variation is around 975 Hz, and cannot be adjusted. The PWM pins may also be operated as ordinary HIGH/LOW.

2.2.2. Programming the UNO



Figure 5 - The Arduino IDE.

*Note: In this manual, Arduino code will be shown **like this**. It may be useful sometimes to copy and paste from the electronic version of this document.*

A detailed description and programming reference is available offline from the IDE's 'Help' menu, or online [4], though the notes in this manual are intended to be sufficient for your purposes. You might find the example sketches built into the Arduino software informative if you're stuck.

You are provided with a USB type A to type B connector cable which allows you both to power up the UNO and to program it. Connect the cable from laptop to UNO and open the Arduino Integrated Development Environment (IDE) software (Figure 5). In this experiment, you will not be expected to write extensive code. The format of the software is very

simple, and is based on the C programming language. Each program, known as a 'sketch', contains two functions (declared by the call `void`, so-named as no value is returned by the function); `setup()` executes first, and just once, and can be used for things like defining the data transfer rate or setting up the pin output, and `loop()` executes infinitely, doing whatever it is the program is designed to do. You may also declare variables or constants before, and outside of, these loops.

Brief Comments on Syntax

As described in the reference material, when programming the Arduino, variables and constants must be declared in a specific way (and usually at the start of the sketch). There are other varieties (see online reference), but those listed in Figure 6 should be more than adequate. You will certainly have problems if your variables are defined or used wrongly, and variables **MUST** be defined before being used for the first time.

```
// note - two forward slashes instructs the Arduino to ignore the rest of
// the line - 'commenting out' user notes.
// note also - this is NOT a useful script!

// logic
boolean myboolean=true;      // true or false

// numeric
int myint=10000;             // 16-bit integer (between -32,768 and +32,768)
long mylong=10000000;        // 32-bit integer (+/-2,147,483,647)
float myfloat=2.67816E21;     // 32-bit floating-point number - 6 to 7
                              // digits precision

//strings & characters
String mystring3="Testing";  // possibly the most useful way of
                              // declaring a string, but the examples below may be useful in some
                              // circumstances. Note the capital 'S'!
char mychar1='A';            // single character
char mychar2=65;             // equivalent, using ASCII value ('A'=65)
char mystring1[7]="Testing"; // mystring1[0] is "T", mystring1[4] is
                              // "i"
char mystring2[]="Testing";  // equivalent, but the UNO works out the
                              // size of the string for you
// note - declare characters in single quotes and strings in double
// quotes.
```

Figure 6 - Some examples of variable declaration for Arduino.

Simple Pinout Examples

Figure 7 shows a simple Arduino sketch to flash the LED near pin 13 on the UNO board. Make sure you understand what each line is doing. Copy the sketch into the Arduino IDE and execute it (note - make sure the IDE is communicating with the UNO ('Tools'→'Port')). Modify it to produce different behaviours.

As well as observing the flashing LED, measure the response on your PicoScope, as measured at pin 13 (remembering about appropriate grounding of your probe). Try other outputs - which pins should you be using?

Now try programming one of the pulse-width modulation (PWM) pins. The command which allows you to write to a PWM pin is `analogWrite(pin,n);`, where 'pin' is the pin number and 'n' is an integer variable between 0 and 255. Observe the output on your PicoScope, and see whether the mean output level is as you would expect.

```
int pin=13;    // declares the variable 'pin' as an integer and assigns
its value

void setup() {    // this loop is performed just once
  pinMode(pin,OUTPUT);    // pin 13 is set up to act as a voltage output
}

void loop() {    // this loop is repeated indefinitely
  digitalWrite(pin,HIGH);    // set pin 13 'HIGH', i.e. 5 V
  delay(1000);    // wait 1000 ms
  digitalWrite(pin,LOW);    // set pin 13 'LOW', i.e. 0 V
  delay(1000);    // wait 1000 ms
}
```

Figure 7 - Arduino LED-flashing program.

Serial Communication

Whilst executing sketches, the UNO is able to communicate with the outside world in real time, and its outputs may therefore be controlled during an experiment. This is accomplished via a serial link. Figure 8 is an example of a very simple sketch which sends and receives strings using this link. Copy and paste it into the Arduino IDE, upload the sketch to the Arduino, and open the IDE's Serial Monitor window ('Tools'→'Serial Monitor'). Type some text into the Serial Monitor. Hopefully you will receive the response you expected. Incidentally, 9600 baud is a rather old standard data rate; we could choose a faster rate, but there would be little point.

You can probably imagine the usefulness of manipulating numbers rather than strings in a physics experiment. Have a go at sending, manipulating and returning numeric data through the serial link, but don't at this stage create anything too sophisticated. The Arduino IDE has many functions for achieving this; feel free (but certainly not obliged) to browse the Arduino reference. You will probably find it useful to use `Serial.parseFloat();` or `Serial.parseInt();` in place of `Serial.readString();`, to read the Serial stream in the appropriate format.

Use what you've learned to make a (very simple) sketch which controls one or more data pin voltages based on what you type into the serial input.

```

int pin=9;      // pin 9 is a PWM output
String message=""; // declare any string variables used later
String reply="";

void setup() {
    Serial.begin(9600); // Initialises the serial link at 9600 bits/sec
                          (9600 baud)
    pinMode(pin,OUTPUT);
}

void loop() {
    if(Serial.available()){ // Checks whether there are any characters
in the serial input buffer
        message=Serial.readString(); // Reads the characters into a
string variable
        reply=String("You typed " + message); // Creates a new string by
concatenating a string and a string variable
        Serial.println(reply); // Sends the new string back via the
serial link
    }
}

```

Figure 8 - Serial communication program.

Speeding Things Up - Arduino UNO Pulse Generator

Your experiment is mainly concerned with the investigation of a digital integrated circuit used for tracking a digital signal, so it is potentially useful to be able to use the UNO to generate a reasonably fast digital, pulsed signal. For now, leave aside the serial communication and write a sketch which will do that, based on the commands `digitalWrite()` and `delayMicroseconds()`; the latter's purpose being reasonably obvious, allowing you to select the frequency programmatically. What can you say about the results you obtain? Any drawbacks or imperfections?

In practice, we need to cheat slightly to get a useful result, and do away with `digitalWrite()`, which is not really designed for this purpose, in favour of some more *direct* coding.

The UNO has three 'ports' (don't worry what this means), of which we will only concern ourselves with port B, which controls pins 8 to 13. All of the pins in this section may be set up to be outputs or inputs, and the register 'DDRB' controls this assignment. The bits in `DDRB`, reading from high to low, represent pins 13 to 8, and the command `DDRB=B010000;` (equivalently, you could use decimal, *i.e.* `DDRB=16;`) sets pin 12 to output mode and the rest to input (the latter 'B' denotes a binary number). To set pin 12 to **HIGH** (5 V), issue the command `PORTB=B010000;` (or decimal `PORTB=16;`). Reasonably obviously, set the relevant bit to zero in `PORTB` to set pin 12 **LOW**. Always set `DDRB` in the setup loop (it needs setting only once), and include with it the very useful `cli();`, which disables the 'interrupt' - the UNO doesn't wander off to check on other processes (try with and without and see what difference it makes). Note the principal features of your results, advantages and drawbacks, and consider how suited the UNO is for acting as a digital pulse generator.

You now know how to program the UNO.

2.3. LabVIEW

LabVIEW is a visual programming language developed by National Instruments [5] principally for the control of instrumentation. It may be used for a wide range of experimental apparatus, and not just NI's own equipment. It is very widely used in scientific research and control engineering. If you pursue a career in experimental physics, it is almost inevitable that you will come across LabVIEW at some point.

Though it is so widespread, and despite being highly visual, it can take a while to get to grips with the basics, and years before the programmer is able to produce something truly robust and, ideally, beautiful. For the purposes of this project, and with a view to not unfairly disadvantaging those with limited or no experience in the area, this experiment will keep it fairly simple. You must remember after all that what you submit should be about the experiment and not the kit.

2.3.1. LabVIEW Basics

LabVIEW in its ultimate state is packaged as a 'project', and may be converted to an executable program for distribution, but at its heart is the 'Virtual Instrument', or 'VI', and much can be achieved with this basic unit.

Open up LabVIEW, and click 'Create Project'. Select 'Blank VI'. You will be presented with a blank 'Front Panel' and a blank 'Block Diagram' (if one or the other is ever not visible, you can open them using the first option on the 'Window' drop-down menu. The Front Panel can be thought of as just that - the front panel of a piece of equipment, on which you can place indicators, switches, graphical output windows and so on. The Block Diagram is where the behind-the-scenes wiring and calculation happens.

To get a feel for using LabVIEW, you will assemble a VI on which you can input two numbers and obtain a result which is either the sum or the difference. Let's start with the Front Panel, adding components one at a time. You should see a 'Controls' window from which you can select items for the Front Panel. If not, open it from the 'View' menu. Navigate into the 'Numeric' folder, click on 'Numeric Control', and place a control wherever you like on the Front Panel. Repeat the process. These items are called controls because they are inputs to the VI; instead, select next a 'Numeric Indicator' (indicator because it displays a result), and place that on the panel. Now we need to place something which will tell LabVIEW whether we wish to add or subtract. This might be a drop-down menu ('Combo Box') from the 'Modern→String & Path' folder (which allows multiple options), but instead use a 'Horizontal Toggle Switch' switch from 'Modern→Boolean'. A 'Stop Button' from 'Modern→Boolean' will be used to exit the program. Your front panel should look something like Figure 9.

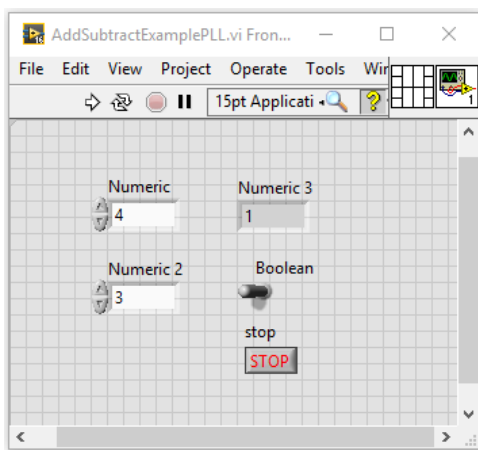





Figure 9 - LabVIEW example Front Panel.

Finally, your boolean toggle switch doesn't show which position does which function, so right-click on it, select 'Properties', tick 'Show Boolean text' and change 'On text' and 'Off text' to say 'Add' and 'Subtract' respectively.

Your front panel is complete; it will look messy, with text in unattractive locations, but you are able to beautify it by showing or hiding various things using the properties menus, by adding decorations and arranging nicely. However, it is quite possible to spend your entire experiment creating something beautiful and space-age, with swishy functions, and it's dangerously addictive. Remember that NO extra marks will be awarded for beauty or unnecessary functionality, and try to proceed with just the basics.

LabVIEW VIs won't do anything until you run them; there's a 'Run' button (the right-arrow  on the menu bar in Figure 9). You'll find that the VI starts briefly, then stops immediately - the VI hasn't been programmed to do anything, so it does nothing. If you select the 'Run Continuously' button, (the circulating arrows  to the right of the 'Run' button), the VI will start repeatedly, allowing you to play with the VI front panel, but of course nothing useful will happen. Click the 'Abort Execution' button  to exit the cycle.

You will hopefully remember that the VI has a 'Block Diagram' associated with it; you need to use this to 'wire up' the controls and indicators to make the VI useful. On the block diagram, you will see five icons, each representing one of the items on the front panel.

The first thing we will do is put everything in a loop, so that the VI can run continuously until the 'STOP' button on the front panel is pressed. When the block diagram is active, you should see a 'Functions' window (if you don't, select 'View→Functions Palette'). Navigate to 'Programming→Structures→While Loop', and, by dragging a selection rectangle, place the while loop around all five of your icons (with a little space to spare). Now, and this will take a little practice, draw a 'wire' from your 'stop' icon to the red stop sign (the loop's exit condition) in the lower right of your while loop (move your icons around first if it's helpful). Your block diagram should look something like that shown in Figure 10 (note - your icons may look different to those shown - 'View As Icon' from Properties menu to change if wished).

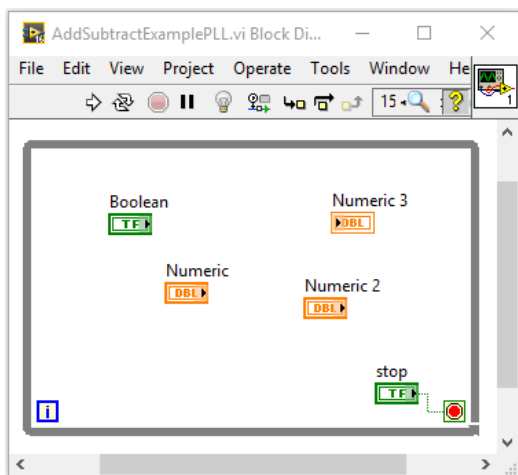


Figure 10 - LabVIEW example Block Diagram.

You may have noticed that in wiring your stop button to the while loop's exit condition, your Run arrow went from broken and greyed to unbroken and white - your VI had errors, but now it's valid. Run it and see what happens, and try stopping with the front panel rather than the terminate button. It should run until you hit 'STOP', but still won't do any calculating.

More wiring needed. We'll move ahead swiftly with the instructions, as you will hopefully be more confident with how this all works. Try to complete the VI by wiring up to create the circuit shown in Figure 11. For this you will need a different type of structure as shown, a 'Case Structure' ('Programming→Structures→Case Structure'). This is a box which (in its simplest form, anyway) executes one of two circuits depending on whether the logic statement wired to it is true or false. Within the case structure, you will need some mathematics. The 'Add' and 'Subtract' functions are available under 'Programming→Numeric'.

Run the VI. You should be able to do all the simple arithmetic you learned as a pre-schooler, but you will also have gained a good understanding of the principles of the LabVIEW Virtual Instrument. Take some time to play with your VI; adapt it to do other things, whether useful or not.

In the next sub-section, you will learn to communicate with real instruments in the execution of a scientific experiment.

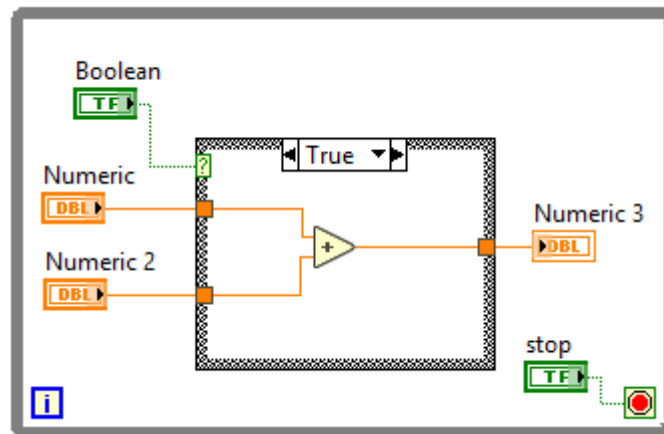





Figure 11 - Completed LabVIEW add/subtract block diagram. The 'False' case (hidden) is identical except it contains a 'Subtract' function.

2.3.2. Communicating with Instruments

There is a baffling array of ways in which instruments are linked with LabVIEW; usually a manufacturer will provide libraries of drivers with functions of varying degrees of user-friendliness, or they may be sufficiently cross-compatible that a generic LabVIEW functions, such as the GPIB interface, are sufficient when coupled with a user manual. It is not possible within the scope of this experiment to study these in depth, so we will concern ourselves only with talking to the Arduino UNO, or to other VIs.

One way of talking to your UNO is using National Instruments' Virtual Instrument Software Architecture, or 'VISA', protocol. In Figure 12 is an example VI which communicates with the UNO, sending a text string and outputting the response. Clearly the Arduino's on-board program can be made to do something with the information it receives, if programmed to do so.

Wire up the VI shown in Figure 12. Some points to note:

- Controls and Indicators are distinguishable on the block diagram, the former having a black arrow (and wire if connected) pointing out of the icon , the latter into it .
- The pink icons are strings, the green boolean and the blue and orange are numeric, denoting different representations (e.g. integer, long etc.). The blue  box is a numeric constant which you may edit if necessary, in this case denoting the (maximum) number of bytes to be read back from the UNO.
- The 'VISA resource name' drop-down can be found in the 'Controls' window at 'Modern→I/O→VISA Resource'.
- The VISA functions on the block diagram are in 'Instrument I/O→VISA' from the Functions window.
- Some inputs/outputs to/from the VISA functions are not connected. For many inputs this means that they take up their default values. Hover over some inputs to see. For example, the VISA serial function has a default baud rate of 9600 - exactly as we chose when programming the UNO.
- The 'False' case of the case structure must be wired also. No functions are necessary - simply wire the purple and yellow/green wires all the way across uninterrupted.
- The green/yellow wire is the error line. It isn't strictly necessary, and it isn't used for any purpose here, but it is good practice to trap errors, and the line can be a handy way of defining the chronological sequence of the operations.
- The purple line passed from function to function is, as it was at the beginning, the VISA resource name, telling the functions which channel to operate on.

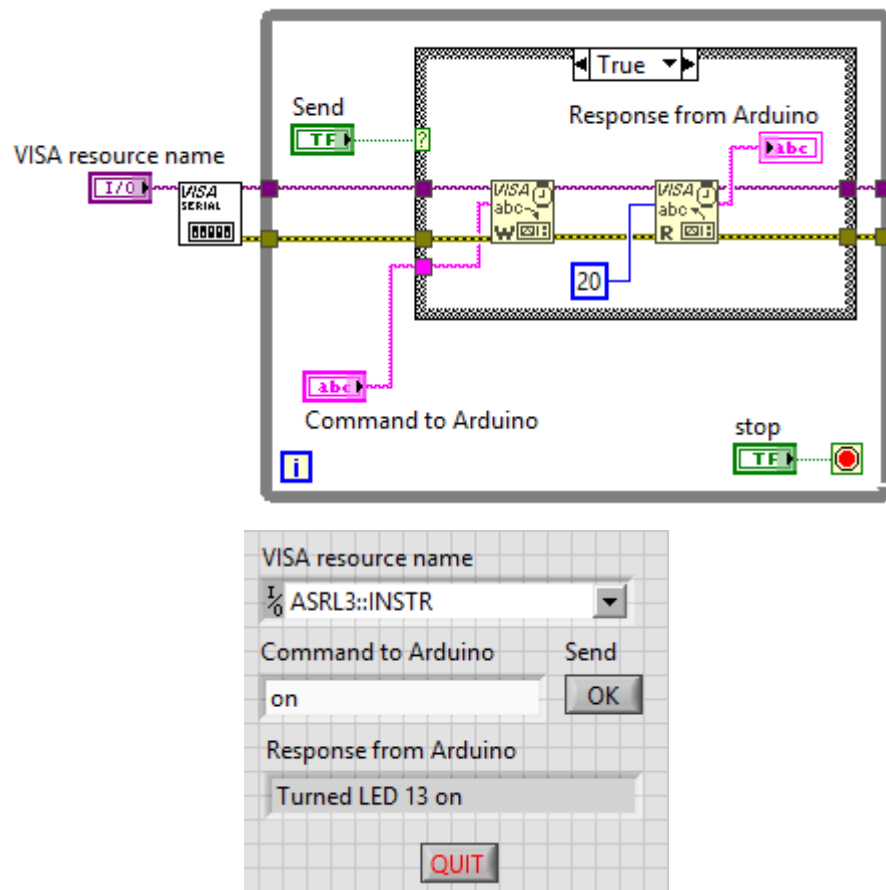


Figure 12 - LabVIEW VI to communicate with Arduino UNO, showing block diagram (upper) and front panel (lower).

Working broadly from left to right, the communication channel defined by the 'VISA resource name' is configured by the 'VISA serial' function. The while loop repeats indefinitely, during which time the case structure box looks for whether the 'OK'/'Send' button has been pressed. If it has, the 'Command to Arduino' string is written to the Arduino by the 'VISA W' function. The 'VISA R' function reads up to 20 bytes from the Arduino and outputs that response into the 'Response from Arduino' string indicator. Eventually when the 'stop' button is pressed, the while loop is exited (*n.b.* counter-intuitively, it seems not to be necessary or useful to use the VISA open and close functions).

Upload a simple sketch to your UNO which listens for a serial input and sends a serial response, then **close the UNO IDE** - you may have hardware conflicts otherwise. Browse for the appropriate resource name - it may depend on your hardware configuration, but is likely to look like that in Figure 12, rather than 'COM1' or 'LPT1'. Check that you can send information to your UNO using LabVIEW, and that you can use that information to control an output pin.

2.4. The Integrated Measurement System

Once you try to incorporate PicoScope into an experimental setup, it will soon become clear that you will want to have a VI to control it, rather than rely on the proprietary software. It is not easy to save experimental data from the PicoScope, and if you are able to do so, it is only the traces themselves which can be exported; none of the measurement functions (frequency, amplitude etc.) that you can invoke during measurement can be saved - you would need to write them down. In particular it is useful to be able to take data repeatedly during a measurement sweep rather than once via the drop-down menus. Finally, it is often a handy thing to be able to synchronise other equipment with the PicoScope so that, say, an oscilloscope trace or measured parameter can be associated with a certain condition on another piece of equipment.

2.4.1. The LabVIEW PicoScope VI

The obvious combination to apply in this experiment is to instruct the Arduino UNO to create voltages or pulses with certain parameters, and measure the experimental output with oscilloscope. Importantly, you will already have discovered that PicoScope is not necessarily perfect, and may not be all that well-calibrated. Measuring what comes out of the UNO with the PicoScope is therefore a sensible way to proceed (if we can assume the PicoScope performs suitably well).

To minimise the often quite considerable time taken in creating a LabVIEW-based experiment, you are provided with two VIs, which can be found on your laptop's Desktop, along with some additional information. Make copies of the VIs as you proceed (and retain the originals) in case you need to revert to previous versions.

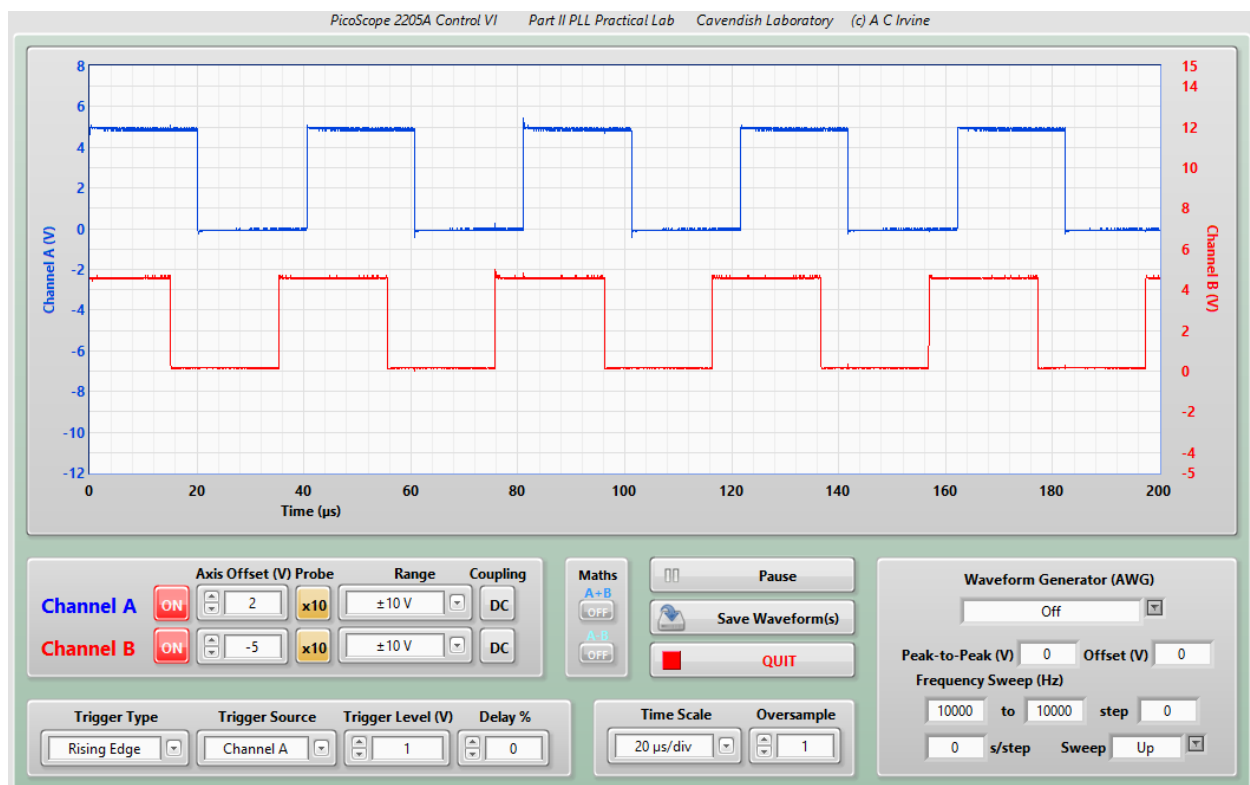



Figure 13 - LabVIEW VI front panel for operation of the PicoScope in a data-acquisition environment.


The first VI, called '**PLL Lab PicoScope.vi**', is designed to replicate the main functions of the PicoScope software but allows simpler data extraction. Its front panel is shown in Figure 13. Run the VI and have a play, to make sure you see what it does. It must be run using the 'Run Continuously' button, , because it is designed to be called by the other VI and to exit once a single pair of waveforms (channels A and/or B) has

been acquired. The VI's block diagram is complicated and you will not need to know anything about its operation (but you are encouraged to take a look if interested). You will NOT be editing this VI!

The second VI is the control VI, '**PLL Lab Control.vi**', shown in Figure 14. The idea of this VI is to deal with all of the data handling and file input/output required by an experiment in which multiple data points (or data sets) are taken.

As currently configured, the control VI records ten rows of data; it requests the waveforms from the PicoScope VI and analyses their data to find their mean values. Data are then sent to the 'Data Out Array' pictured at centre. In the example, the first two columns are elapsed time in seconds and row number (starting at zero). The fifth column (and beyond) is not defined, but can be later.

The upper panel shows waveforms in real time with auto-scaling, and there is a display space set aside to show your calculated data in real time; this may be customised as you like, but is optional.

Hopefully the buttons are obvious. The 'RUN' button starts taking measurements, the 'CLEAR' button empties the data array and resets the data count to zero, and the 'SAVE' button opens a dialogue with options for where (and in what form) to save the array data. Finally, the 'QUIT' button terminates both VIs (though no harm will come if you do this by hitting the main LabVIEW 'Abort' button, ).

The block diagram for this VI needs some thought. Look at Figure 15 and make sure you have a good idea how this VI controls the experiment and accumulates data. In particular, note regions (h) and (l) on the figure; in (h) you may need to set some experimental parameters, such as telling an Arduino what to do, before recording the data. This may well involve looking for 'Select a VI...' on the Functions Palette. Be warned - make sure the PicoScope VI takes data AFTER the experimental conditions are set - wiring the 'Error Out' of the relevant VI to 'Error In' of the PicoScope VI icon is a way to ensure this.

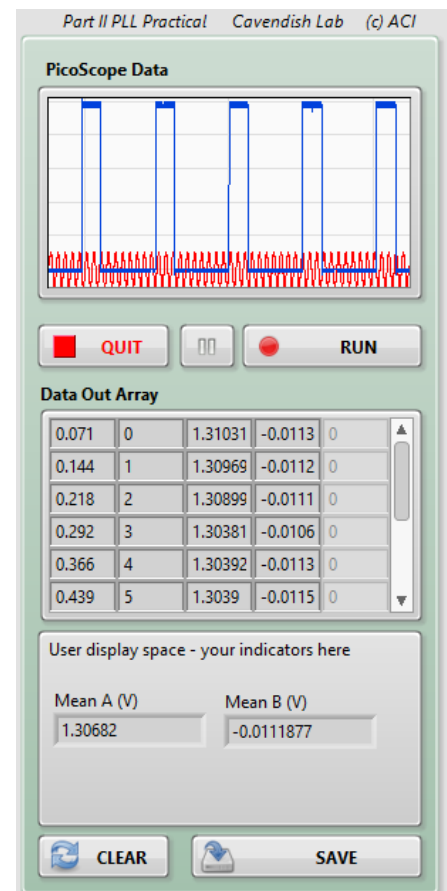


Figure 14 - LabVIEW VI front panel for controlling experiments.

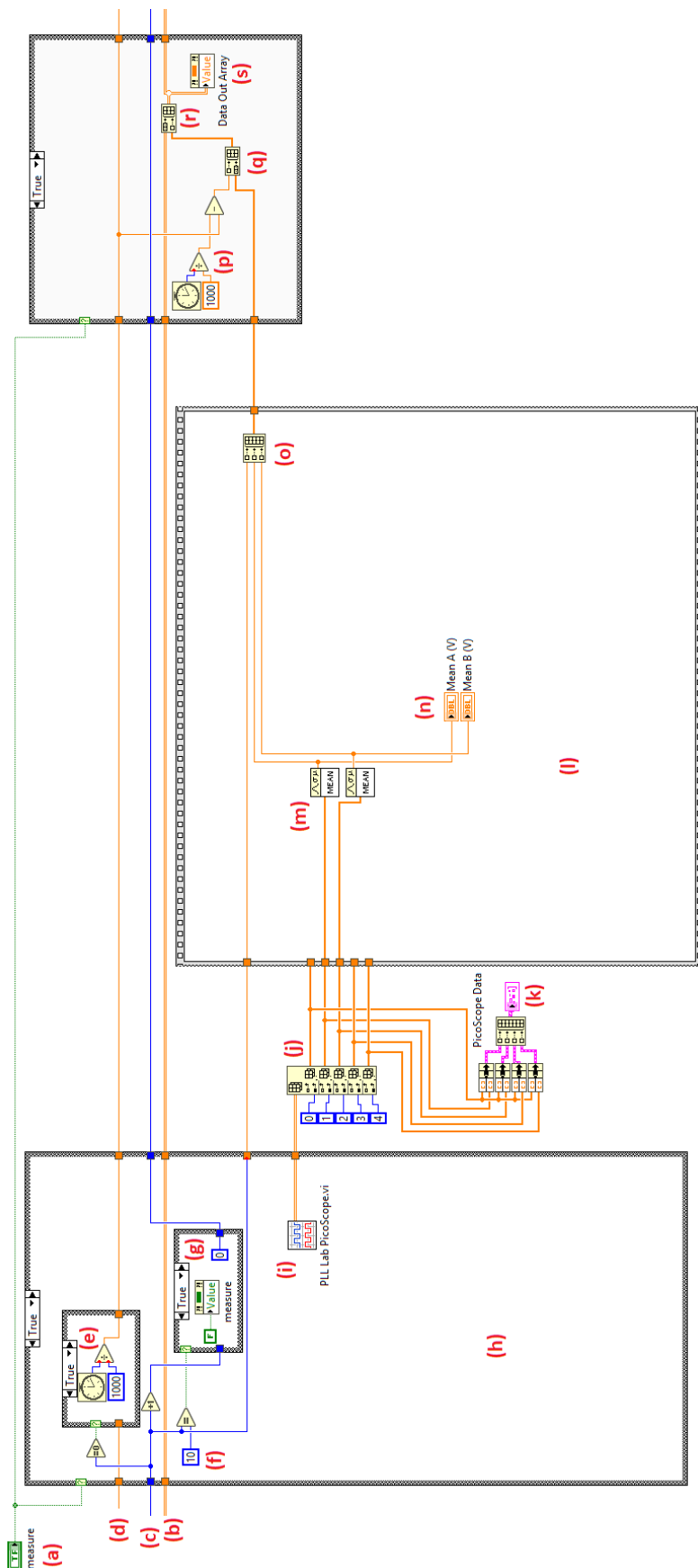


Figure 15 - Detail of data-processing part of the control VI block diagram. A boolean icon (a) represents the 'RUN' button - if pressed, the bulk of this VI section is executed. At (b) is the data 'wire' which represents the measured data array, at (c) is the data count, to be incremented from zero in unity steps, and the wire at (d) is the absolute time at which the measurement commences; if the data count is zero, this time is recorded using the True/False case structure (e), otherwise it is unchanged. (f) is the data count of the final data set to be recorded in each sweep (e.g. to sweep 0 to 5V in 1V intervals, this number should be 5). Once this value is equalled, the 'RUN' button is switched off (g) and the number count reset, stopping data accumulation. A region here is left blank (h) for the programmer to set one or more parameters of the experiment, such as setting a bias voltage. Values for such parameters may be obtained from the data count, and may be passed on to the measurement part for inclusion in the data array. Regardless of whether data is being recorded, the PicoScope VI is invoked (i) and waveform traces recorded as a 2-D array. This array is then broken into (1-D) columns (j), where column '0' is the time and columns '1', '2', '3' and '4' are the A and B channel voltage and their sum and difference respectively. The three 1-D arrays may then be processed by the programmer in region (l) (note that the square 'Flat Sequence' container is unnecessary, being included only for clarity). An example analysis (m) takes the mean values of the two voltage arrays and displays those values in the (optional) indicators (n) described earlier. At (o), a 1-D array is created with, in this case, three values (data count, mean A and mean B) being placed in the array using the 'Build Array' function block (to incorporate more variables, enlarge this block). Finally, the elapsed time is calculated (p) and placed at the start of the data array (q). The completed array is added as the last row of the 2-D data array (r), and the result placed in the 'Data Out Array' indicator (s).

One last thing - the PicoScope VI allows you to control some aspects of the arbitrary waveform generator (AWG). You might find it useful to control the AWG during a programmatic sweep. One of many ways of doing this is to find the 'PLL Lab PicoScope.vi' icon, at point '(i)' in Figure 15, and follow this sequence:

- Right-click on the upper left side of the icon, at the terminal called 'AWG Cluster' and select 'Create→Constant' from the menu; a Cluster in LabVIEW is a grouped set of variables, here representing the AWG controls.
- Drag all of the variables outside the cluster box, keeping in order, and delete the box and wire.
- Place a 'Bundle' function from the 'Programming Cluster, Class, & Variant' palette, and wire each variable in turn to it. Feed the bundle output into the 'AWG Cluster' input of the 'PLL Lab PicoScope.vi' icon.

The resulting circuit should resemble Figure 16, and you will be able to change or perform logic operations on any of the AWG inputs.

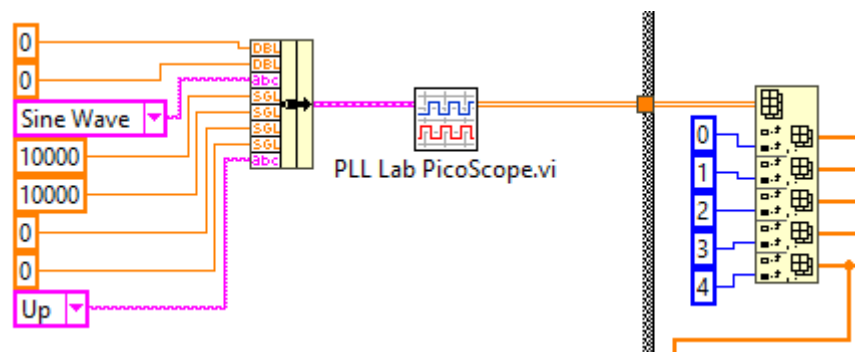


Figure 16 - Detail of the Control VI, with the addition of code to control the arbitrary waveform generator.

This will provide a relatively simple way of applying an arbitrary frequency to your experiments, at least up to the AWG's maximum value of 100 kHz. However, the AWG is limited to the range ± 1 V, and that entire range is recognised as logic zero in 5 V digital electronics. You will need to produce a logic-level signal either using an LM393 comparator (Figure 41), which produces logic '1', +5 V, for inputs above 0 V, logic '0' otherwise, or by configuring a suitable voltage amplifier (Appendix A4).

All that remains is to record some data. In this experiment you will investigate the operation of a 4046-type phase-locked loop integrated circuit. For each part, it is up to you to modify the control VI appropriately.

Background to the Experiment

3. Introduction to Phase-Locked Loops

A phase-locked loop circuit employs feedback to maintain a local oscillator in synchronism with an incoming signal. Its wide applications include signal recovery and demodulation, and frequency multiplication. This experiment investigates the operation of the circuit and the feedback conditions necessary to achieve both stable operation and good response to changing input signals, and considers examples of the applications for the circuit.

3.1. The Experiment

The experiment involves the assembly of electronic circuits which allow investigation of an integrated circuit, the HEF4046B chip ('4046' for short). Circuits will be constructed on a prototyping board, consisting of a 'breadboard' - a push-fit matrix with interconnecting tracks - and a power supply, upon which circuits may be assembled and modified non-permanently. In some cases, the circuit will be complicated, and will require care and attention to detail. It is generally very worthwhile considering carefully the positioning of the circuit before starting, and taking steps to make the circuit easy to debug by both experimenter and demonstrator, for example by using a colour code for the wiring and by using appropriate lengths to avoid a 'rat's nest' of tangled wires. Traces will be observed on a PicoScope using LabVIEW or, if appropriate, the proprietary PicoScope software. This is the first year of an upgraded method, so if you have any teething troubles it is quite possible that the experiment itself is flawed, so please raise the issue straight away. Basic instructions for the breadboard are given in Appendix A1. Whenever in doubt, *ask*.

A considerable part of your lab time will be taken up in assembling these circuits correctly. However, the class is not an exercise in the completion of the greatest number of manual tasks in the given time; the time taken once the circuit is assembled, when you will be studying the behaviour of your circuit and its optimisation (and corresponding write-up time), is by far the most important time you will spend. It is by no means a problem if you do not complete all of the applications in this manual. A report with good experimental techniques and careful analysis based on a subset of the available tasks will attract far better marks than one which merely ticks the boxes - getting all of the circuits to work with no further thought. As an indication, good marks can be obtained by completing Section 7. If your experimental approach and analysis is of a high standard, but the best marks will be gained by, in addition, covering some of the applications in Section 8.

This manual gives the principles of phase-locked loops in Section 4 and specifics of the applications in Section 5. Your first experiments will come in Section 7, but you would be well advised to be familiar with the principles before plugging in your first component.

Finally, you are learning the art of experimental work as well as its science. If you have a bright idea for a brief experiment which is both relevant and potentially informative, feel free to include it.

4. Phase-Locked Loop System Operation

4.1. Feedback and Stability

The phase-locked loop circuit is a feedback system, in which comparison is made between the incoming signal and the output signal from the voltage-controlled oscillator to produce an 'error' signal which represents the phase difference between them (Figure 17). This 'error' signal is applied to a voltage-controlled oscillator to determine its output frequency. It is often necessary to incorporate a filter circuit, and the characteristics of that filter must be chosen to remove unwanted signal-frequency (and other high-frequency) components from the error signal and to ensure stability and satisfactory response from the system. The filter's optimum response time will be application-dependent, and is a key parameter in successful operation. The 'error' signal may in some cases pass through an amplifier (not shown) before insertion into a voltage-controlled oscillator (VCO).

In some applications, the filtered 'error' signal is itself the required output from the phase-locked loop. For example, in frequency-modulation detection the 'error' signal is a measure of the frequency of the incoming frequency-modulated signal, and so gives the modulation signal that is to be recovered by the detector.

Analysis to determine the stability of the circuit is given in Appendix A2. It shows that the basic system, without any filter, is inherently stable.

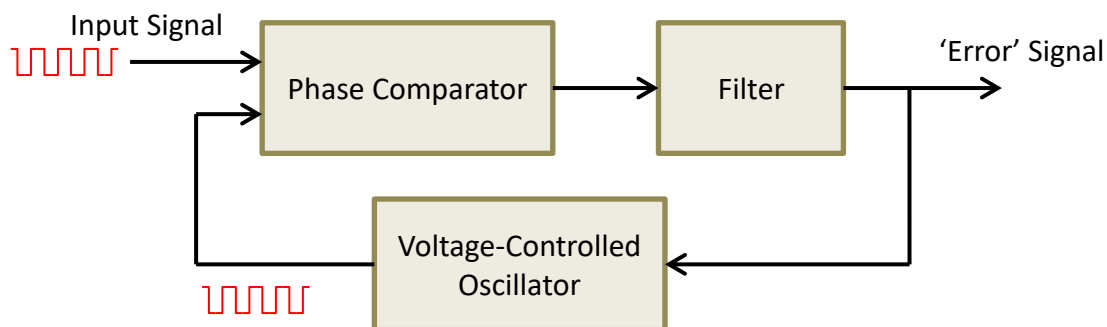


Figure 17 - Basic elements of a phase-locked loop feedback system.

4.2. Effects of Filter Circuits on Stability

With no filter present, the phase-locked loop circuit is very limited in its applications. Real comparator signals vary over a cycle of the signal frequency, and this variation may need to be smoothed out. Additionally it may be required that the voltage-controlled oscillator provides a steady output even if the input signal is subject to noise or has missing cycles (in clock-signal regeneration, for instance) or if the frequency of the signal applied to the comparator is different from the frequency of the oscillator (as in a frequency multiplier).

In such cases, the obvious approach is to interpose an R-C integrator (Figure 18) to smooth the 'error' signal and to apply the smoothed signal to determine the frequency of the voltage-controlled oscillator.

However, this filtering can affect the stability, as discussed in Section A2; to overcome this problem a variant on the basic integrator can be used, the lead-lag circuit of Figure 19.

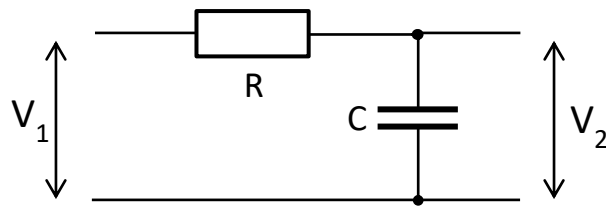


Figure 18 - Integrator using a simple R-C filter.

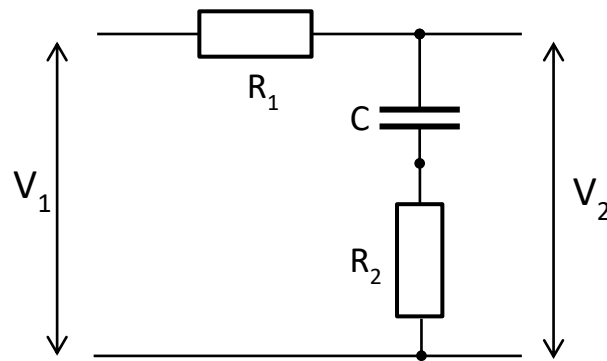


Figure 19 - Integrator using a lead-lag circuit for zero phase shift at high and low frequencies.

4.3. Tracking and Capture

Once the oscillator has locked to the frequency of the input signal, the oscillator frequency will track the input provided that the rate of change of input frequency is not excessive. For systems in which the control voltage for the oscillator does not vary significantly during any cycle of the oscillator output, tracking will occur provided the input signal remains within the range that the voltage-controlled oscillator can provide for the available control voltage swing; if the control voltage varies over a cycle, more restrictive conditions may apply.

The capture process by which the local oscillator initially locks onto the incoming signal is relatively complicated, and capture is not necessarily possible for the full range of frequencies within which tracking is possible. Both the oscillator frequency in the absence of signal input and the capture characteristics depend on the type of comparator.

5. The CMOS Integrated Circuit Type 4046 Phase-Locked Loop

The venerable HEF4046 phase-locked loop chip has long been at the heart of this experiment. However, there is a long tradition of CMOS manufacturers releasing new generations which are functionally identical to their predecessors. In the case of the HEF4046, we can therefore substitute with the 74HC4046, which has all the same pin-out functions as the 4046, with just one small difference - a third type of phase comparator output, replacing the HEF4046's Zener output on pin 15. We will not be investigating this third comparator this year.

There *should* be enough HEF4046s to go round this time, and I suggest you use them in case of unforeseen issues. On the other hand, I would be interested to see whether the new version performs relative to the old, so the brave may opt straight away for the 74HC4046. In the event of all of you needing to upgrade, everything will be fine. Or if it isn't, either raise the issue straight away or, again if you're brave, characterise the difference.

5.1. General Outline

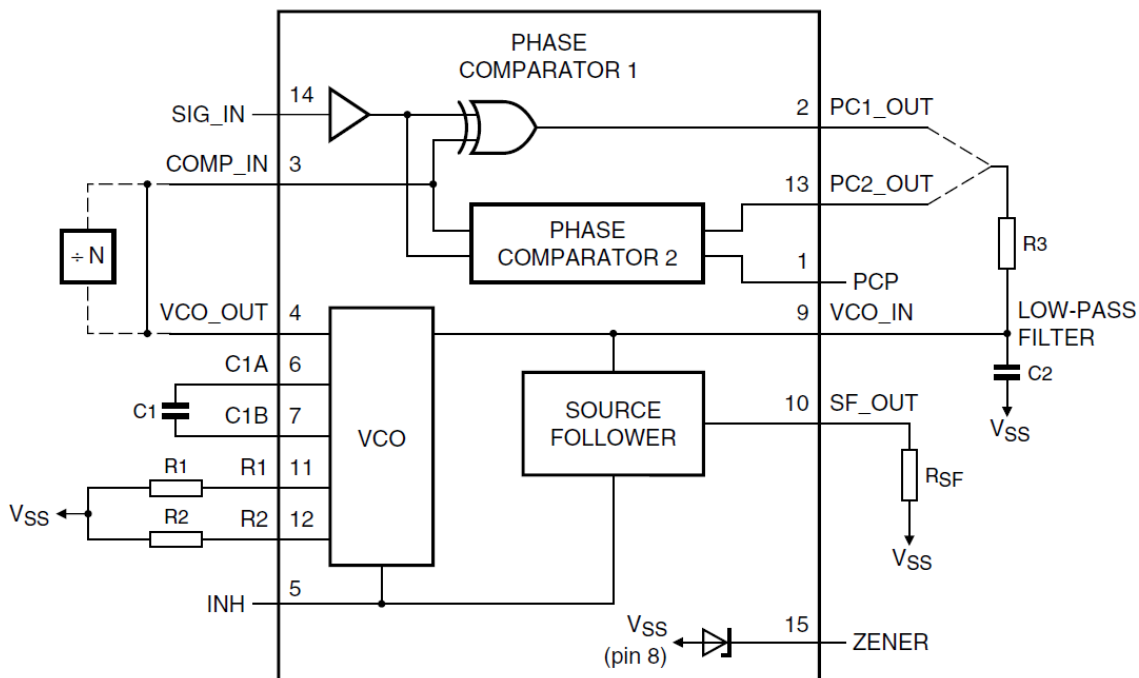


Figure 20 - Schematic of the HEF4046B phase-locked loop integrated circuit [6].

The 4046 CMOS (complementary metal-oxide-silicon field-effect transistor) integrated circuit incorporates all necessary active elements for a phase-locked system, requiring the addition of only a few passive components. It contains a voltage-controlled oscillator and two different phase-comparator sections, with common input circuits (Figure 20). The input amplifier for the 'signal' channel can accept small signals as well as CMOS logic-level signals, but for this experiment only logic-level signals will be required. With a supply voltage of 5 V, logic level '1' is about 5 V and '0' about 0 V. Pin connectors are given in Appendix A3.

5.2. Phase Comparator Type 1: EXCLUSIVE-OR Circuit

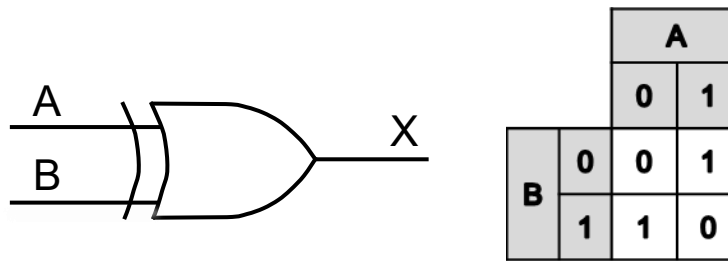


Figure 21 - EXCLUSIVE-OR circuit symbol and truth table.

The EXCLUSIVE-OR circuit provides an output which is high when only one of the inputs is high, so that the truth table for the output X is as shown in Figure 21.

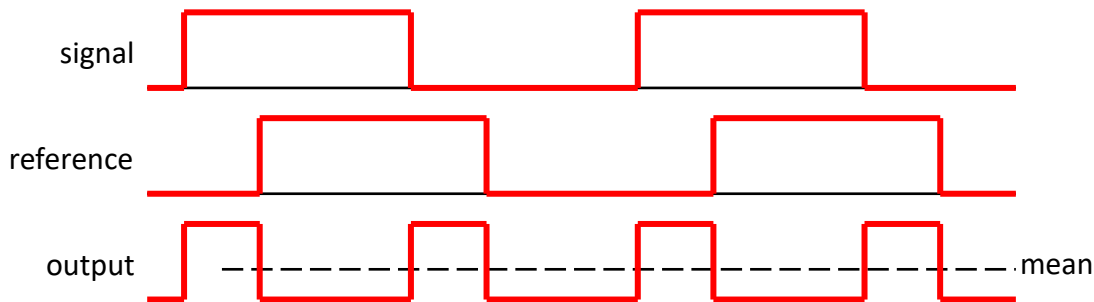


Figure 22 - EXCLUSIVE-OR circuit: idealised signals.

With two identical input signals the output is always low, whilst two signals with the same frequency but different phase will give a rectangular-wave output signal (Figure 22). This signal has twice the frequency of the input signals, with its mark-space ratio dependent on the phase difference; thus the mean level of output is proportional to the phase difference between the inputs. Note that when the mean output level is at its mid value (because its mark-space ratio is unity) the input and the reference signals are 90° out of phase.

5.3. Phase Comparator Type 2: Edge-Triggered Circuit

The second phase comparator produces output pulses which depend on the rising and falling edges of the input reference signals. These pulses are intended to be applied to an R-C integrator circuit, external to the 4046 chip, so that the voltage on the capacitor can be used as the input to the voltage-controlled oscillator. When the input and the reference signals are at the same frequency (with phase difference constant), the capacitor voltage must be held constant; for the complete phase-locked loop, this voltage would be the voltage for which the oscillator frequency is equal to the input frequency. If the input and the oscillator signals are at different frequencies, the capacitor voltage must be driven in the direction that would bring the oscillator into synchronism.

The requirement for increasing the integrator voltage, or for decreasing it, or for maintaining it constant is achieved by driving the R-C integrator from a three-state output stage by which the capacitor voltage can be driven higher or lower, or the capacitor can be isolated (Figure 23). The state of the output stage is determined by a logic circuit whose operation is indicated by the state diagram in Figure 24; each circle in this diagram represents a state that the logic circuit can occupy, whilst the positions of the circles and their shading indicate the corresponding states of the output stage. Take some time to understand Figure 24 fully.

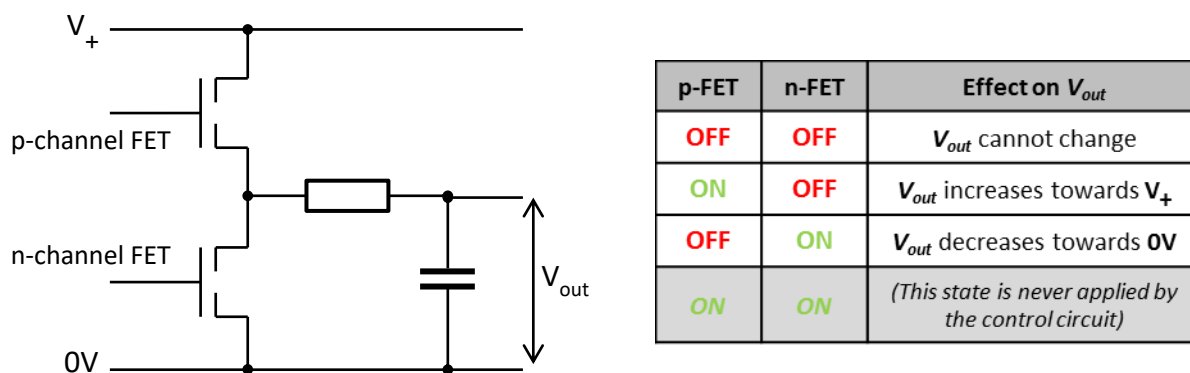
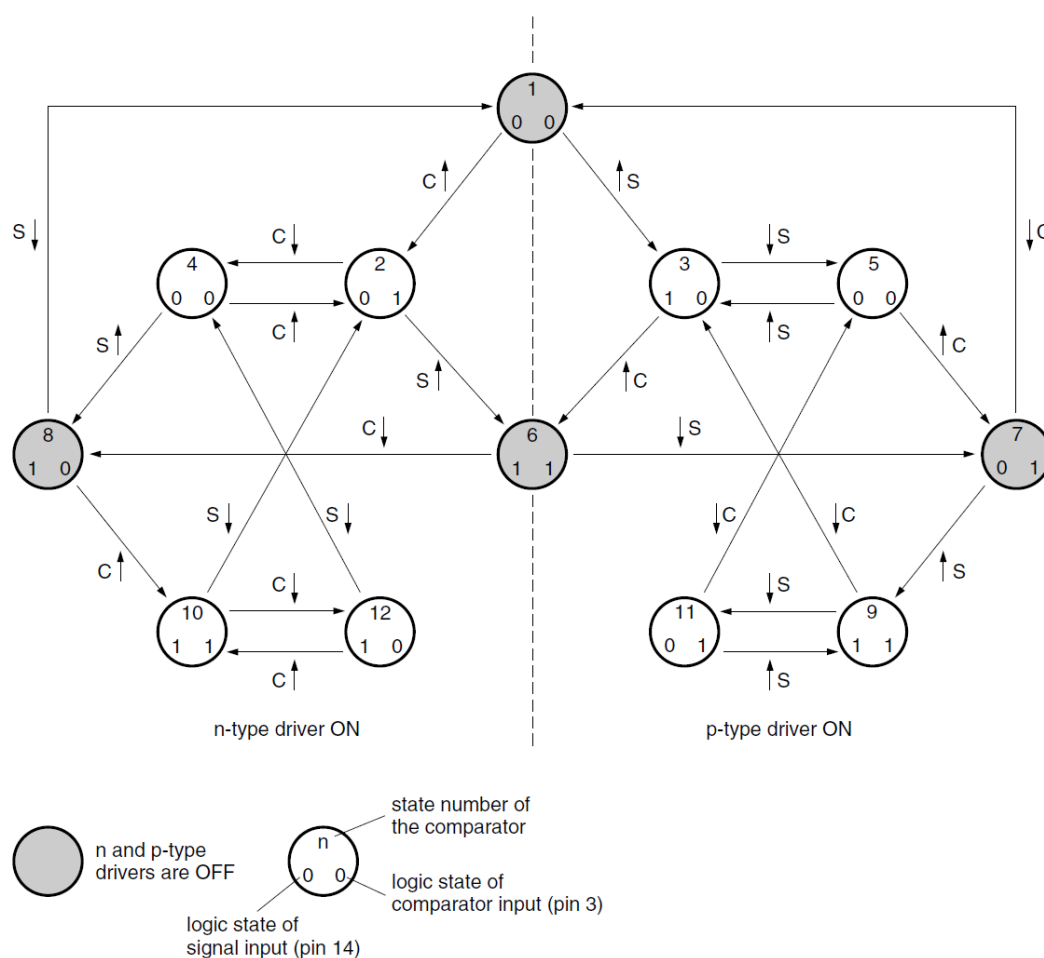


Figure 23 - Schematic of three-state output stage driving an R-C integrator; in the 4046, the two FETs are incorporated in the integrated circuit, whilst the capacitor and the resistor are external.



S ↑: 0 to 1 transition at the signal input SIG IN.

C ↓: 1 to 0 transition at the comparator input COMP_IN.

Figure 24 - State diagram for type-2 comparator in Philips HEF4046B. n- and p-type 'drivers' refer to FETs of the type shown in Figure 7; both are 'off' for the states shaded grey, whilst the unshaded states in the left and right halves represent, respectively, n-FET 'on' and p-FET 'on'. [6]

Transitions between the logic-circuit states occur according to the rising and falling edges of the input and reference signals. For example, if the input is at the same frequency as the reference signal, and leads it

in phase, then their edge transitions will follow the cycle $C\downarrow, S\uparrow, C\uparrow, S\downarrow, C\downarrow, \dots$ and the corresponding state-number sequence would be 7, 1, 3, 6, 7, so that the output stage would have both transistors off except for the period between $S\uparrow$ and $C\uparrow$, during which the integrator would be driven positive, thus tending to increase the local oscillator frequency and so tending to pull it into phase with the incoming signal. In general, the edge transitions follow a sequence which results in output-stage transistors being turned on to give an appropriate change to the integrator voltage; for instance, if the signal frequency is initially four times the reference frequency, a cycle such as $C\downarrow, S\uparrow, S\downarrow, S\uparrow, S\downarrow, C\uparrow, S\uparrow, S\downarrow, S\uparrow, S\downarrow, C\downarrow, \dots$ might result, with a state-number sequence 11, 5, 3, 5, 3, 5, 7, 9, 11, 9, 11, which would hold the p-channel FET on for most of the time, tending to drive the integrator voltage higher. From arbitrary starting conditions the state sequence will be less regular, and will change as the oscillator frequency is pulled into synchronism.

The HEF4046 and 74HC4046 chips both operate in the same way, though the data sheets may seem different.

5.4. Voltage-Controlled Oscillator

The voltage controlled oscillator in the 4046 uses drive circuits similar to CMOS output stages to drive an integrator, so that switching occurs when the voltage of an external capacitor C reaches a particular value. It is probably similar to Figure 25.

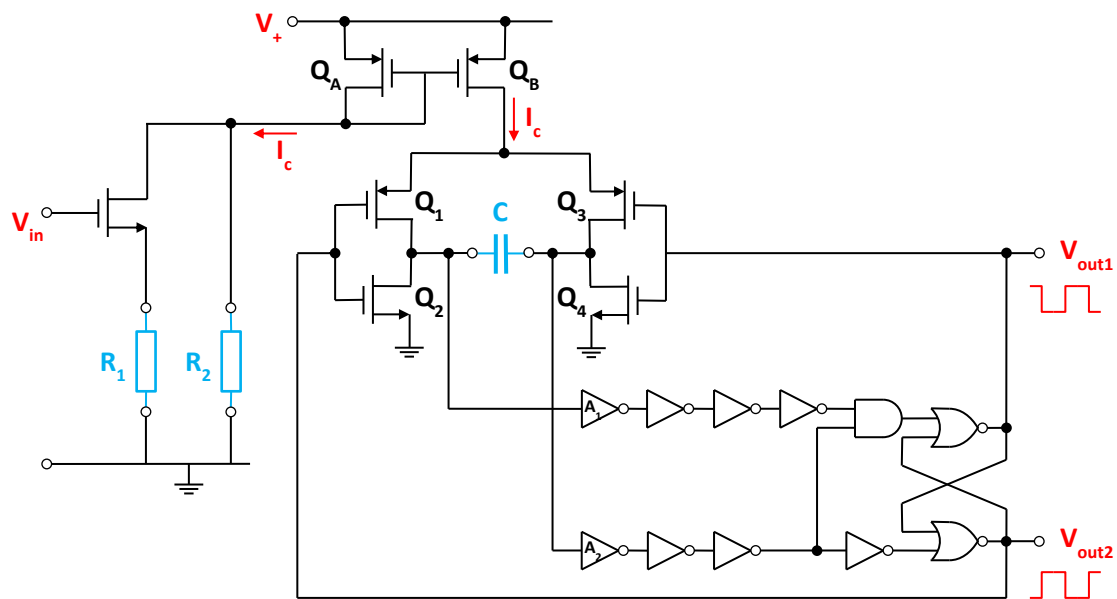


Figure 25 - Plausible arrangement of the voltage-controlled oscillator. C , R_1 and R_2 , highlighted, are discrete components external to the integrated circuit.

The capacitor is charged alternately with opposite polarities. When the output voltage reaches a reference value, one of the comparator circuits A_1 and A_2 trips, and the charging current drive is reversed. The input stage is a current-mirror circuit so that (if the two FETs Q_A and Q_B are identical) the charging current I_c will equal the input current. As shown, the total input current is the sum of the two terms, a fixed term set by the external resistance R_2 and a variable term set up by V_{in} and the resistance R_1 .

5.5. Quiescent Conditions and Signal Capture

The two types of comparator behave differently both in the absence of an input signal and in their response when a signal is applied. In some cases the sudden application of too high an input frequency can result in the local oscillator running at a sub-multiple of the input frequency rather than locking into synchronism; this can occur even for a frequency at which the oscillator would operate in synchronism if the input frequency had initially taken a lower value and had then been raised sufficiently slowly. Thus the frequency range over which the signal capture can occur is not necessarily the same as the range over which oscillation is possible, and the local oscillator may fail to track the input signal.

For the type-1 comparator in the absence of input signal, the exclusive-or gate output is the same as the reference (local-oscillator) signal. If the local-oscillator mark-space ratio is unity, then the mean comparator signal is mid-way between the limiting values, and thus with the feedback loop closed the local oscillator frequency is driven to the middle of its frequency range (neglecting non-linearities).

In the corresponding case for the type-2 comparator, only reference (local-oscillator) edge transitions are present. This is equivalent to the reference frequency being very much greater than the input signal frequency (and so with the logic circuit states alternately 2 and 4, or 10 and 12), and thus the local oscillator frequency is driven to its lower limit.

The type-2 comparator permits capture over the full frequency range of the oscillator, as can be seen from consideration of the logic sequence in Figure 24. For an input frequency which is several times greater than the oscillator frequency. It provides a drive signal which would tend to drive the oscillator frequency in the required direction, irrespective of whether (or how rapidly) the oscillator responds.

However, the use of the type-1 comparator can enable the oscillator to lock onto a signal in such a way that the input and oscillator frequencies are harmonically related, and thus capture over the full oscillator range may not be possible.

There is a further problem with the type-1 comparator. Consider the operation of the circuit with an integrator with a very long time constant. Before the feedback loop is closed, the detector output will have a mean value of one half the peak level, irrespective of the frequencies of the two square wave signals, if it is averaged over a sufficiently long period. Therefore when the feedback loop is closed the input to the voltage-controlled oscillator will remain constant and capture will not occur. Thus for capture to occur, the filter time-constant must be sufficiently short for the local-oscillator frequency to change within the period over which the comparator output varies.

5.6. Noise and Interference

Susceptibility to noise or interference is a function of the comparator type. With the type-1 comparator, the effect is immediate and occurs only during the presence of the interfering signal. The type-2 comparator can take some periods to recover from interference, and this can be shown by considering the output waveform from the logic of Figure 24 with different starting conditions.

6. Integrated-Circuit Layout and Operation

The 4046 phase-locked loop integrated circuit is currently in production, despite being quite an old design (the original 4000-series CMOS circuits were first introduced in about 1980). In consequence, the individual elements of the integrated circuit (transistors, conductor tracks etc.) are quite large and can be seen readily under an optical microscope. For instance, the minimum conductor-track width in the circuit is about $5\text{ }\mu\text{m}$ (for comparison, modern high-performance mass-produced processor chips have feature sizes approaching 10 nm).

Figure 26 is an optical micrograph of the circuit chip, which is actually about $1000\text{ }\mu\text{m}$ by $1200\text{ }\mu\text{m}$. It may be useful to relate some conclusions to this figure. If you are interested, some tips for recognising components of the chip may be found in Appendix A4.

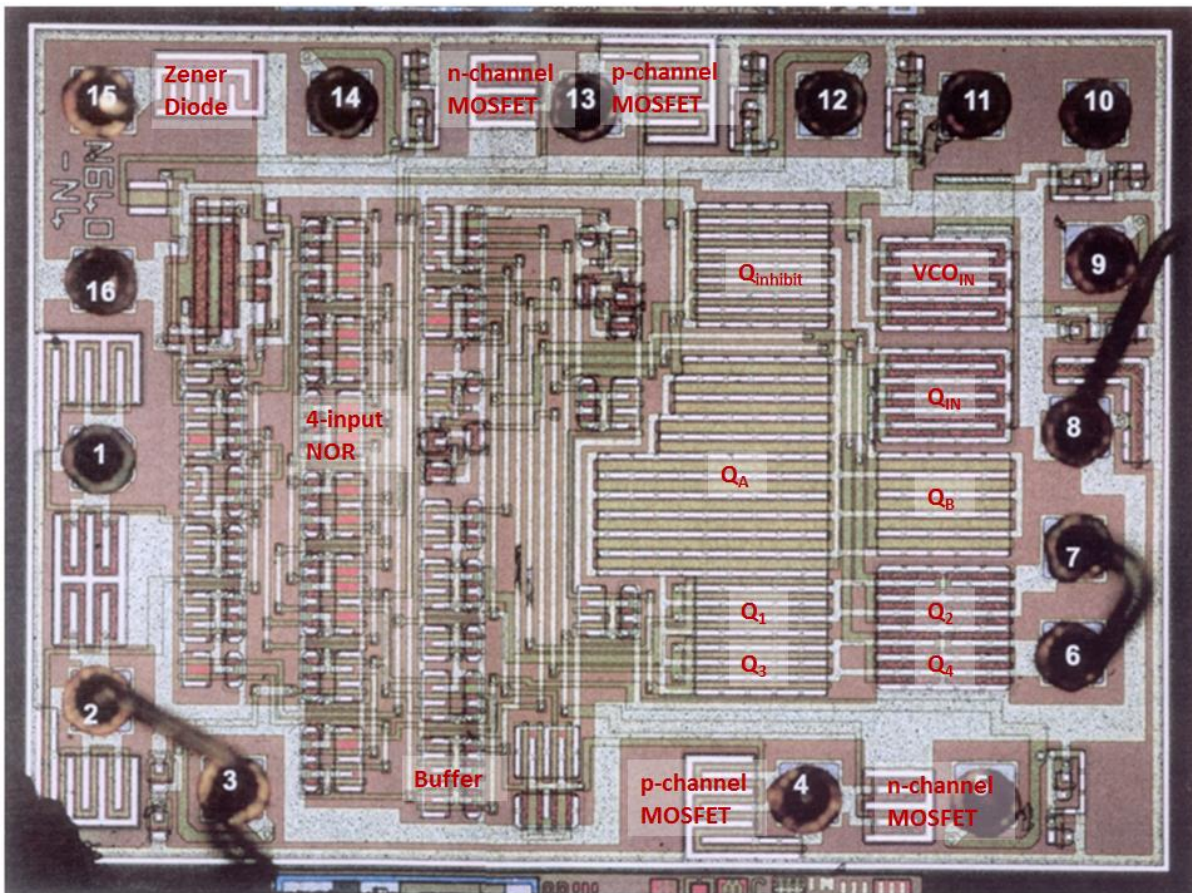


Figure 26 - HEF4046B chip annotated to show key areas, with pins numbered as in the pinout diagram (Figure 20). The logic circuits make up the bulk of the left-hand side. See elsewhere in this manual for description. Chip reverse-engineered by D. J. Weaver [7].

The Experiment

7. Experimental Characterisation of the Phase-Locked Loop Integrated Circuit

The characteristics of the elements of the 4046 CMOS integrated circuit should be investigated and these elements combined to form a complete feedback system. Response and stability should be investigated for the basic phase-locked loop, which is then adapted to meet specific functional requirements.

Even when a method is suggested in this manual, it is up to you to decide how best to approach the experiments. You have the electronic apparatus discussed earlier, but feel free to use the signal generators provided, or to build any circuitry, or to make use of the AWG function of the PicoScope VI. You may wish to take steps to smooth the signal using suitable capacitors to ground.

Since you will have differing backgrounds and skills, you may approach the experiment according to your abilities; some may focus on sophisticated programming and analysis, some on their insight into analogue or digital electronics. Programmers should remember that no credit is obtained by applying flashy features which reveal nothing about the system under test, and electronics buffs should bear in mind that this experiment should make use of automated measurement.

However you approach it, you must always remember to apply the usual rules of scientific experimentation; conclusions must be backed up by the available data, and validity demonstrated. This should be a characteristic shared by all.

7.1. Phase Comparator Characterisation - Phase-Shift Circuit

To demonstrate the phase-comparator characteristics, it is necessary to have two signals at the same frequency with a controllable phase difference between them. The use of separate oscillators does not provide adequate control. To facilitate your measurements, you should create a phase-shifter instrument by which it is possible to shift the relative phase of the signals in either direction, passing smoothly through the condition in which the two signals are in phase. The practical circuit should meet another requirement: for compatibility with the CMOS logic circuits: it should provide square-wave outputs, switching between logic levels '0' and '1' (here about 0 V and about +5 V).

Hopefully you will have spotted that your Arduino UNO boards provide a possible means of applying a suitable signal. Prepare a suitable Arduino sketch, either in association with a LabVIEW measurement or not, to provide a variable phase difference between two logic-level square-wave signals, evaluate its performance and discuss any drawback you encounter.

A general constructional point, to be considered here and throughout the experiment, is the need to ensure that the power-supply lines do not provide unwanted signal paths. Particularly (but not solely) for fast digital circuits, rapid change within an integrated circuit can cause sudden change in the current drawn from the power-supply line and thus (since it is not supplied from a zero-impedance source) in the supply-line voltage. This voltage change can couple into other circuits, causing spurious operation. The cure is to improve power-line decoupling: in addition to putting a capacitor across the supply at its entry to the circuit board, it is appropriate to put a low-loss capacitor (ceramic or polyester) between each supply rail and ground in the immediate vicinity of each integrated circuit.

7.2. Phase-Comparator Characterisation - Type 1 Comparator

Apply the signals from the phase-shift system to the signal and reference inputs of the phase comparator, and record representative waveforms. What is the transfer characteristic of the comparator, that is, the dependence of the mean output voltage on the phase difference between the inputs?

7.3. Phase-Comparator Characterisation - Type 2 Comparator

To demonstrate the characteristics of this comparator without introducing ambiguity due to the high-impedance state of the three-state output (which is part of the integrated circuit), connect the output to a suitable reference voltage from a high-resistance potential divider, Figure 27. Then the signal monitored at the PicoScope will be held at 'logic 0' or at 'logic 1', provided that one of the transistors is turned on; if both are turned off, then the observed voltage will be that given by the potential divider. Record representative waveforms, and estimate the transfer characteristic for the dependence of the mean output voltage on the phase difference. Potential divider resistances should be chosen to give the expected behaviour, and should not be retained in later circuits.

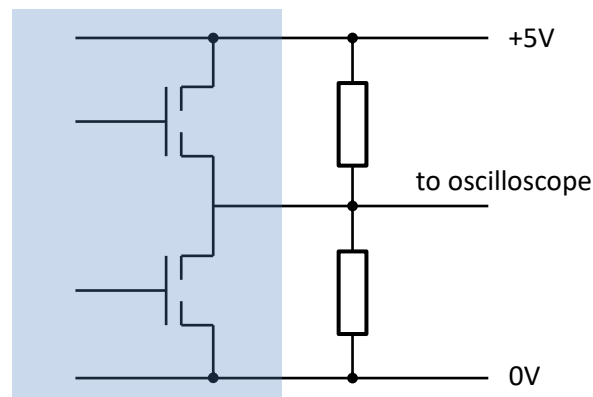


Figure 27 - Load circuit to show the output characteristics of the type-2 comparator. Importantly, the area shaded in blue is part of the phase comparator's on-chip output circuitry, and the transistors are controlled by internal logic (the nature of which is beyond the scope of this experiment); only the resistors are added externally, and they should be removed in subsequent experiments.

7.4. Voltage-Controlled Oscillator Characterisation

The oscillator in the 4046 integrated circuit permits independent control for the centre frequency and for the frequency range that can be covered. This is represented in Figure 25, which is a plausible schematic for the circuit.

What is it about this circuit which makes it a 'current mirror', constrained to supply identical currents to its two branches (if we assume that Q_A and Q_B are identical)?

The oscillator frequency is directly proportional to the drive current to the capacitor, and that current depends on two resistors and the control voltage. The resistor R_2 sets a standing value of the capacitor drive current and so sets the frequency of the oscillator when the control voltage is zero. The load resistor R_1 for the input source-follower sets the relationship between the control-voltage increments and the drive-current increments. To avoid excessive input currents, neither R_1 nor R_2 should be less than 10 k Ω .

Because the integrated circuit operates with a single power supply of +5 V, it is convenient to choose the free-running frequency to be the frequency when the control voltage is half the supply voltage (+2.5 V) and so to allow for frequency excursions in either direction. This is consistent with the use of the EXCLUSIVE-OR

comparator: with input and oscillator signals at the same frequency and with relative phase $\pi/2$, the comparator will provide a 5 V peak-peak square wave having a mean value of 2.5 V.

Set up the voltage-controlled oscillator with $R_1 = 100\text{ K}\Omega$, $R_2 = 100\text{ K}\Omega$ and $C = 1000\text{ pF}$, and check that the voltages on each end of the capacitor are consistent with the assumed form of the circuit. Does the current-mirror circuit give the same current in each branch (*i.e.*, in the input side and in the capacitor side)?

Explore the dependence of the oscillator frequency on the value of the capacitor and the resistors (for a few values, say $10\text{ K}\Omega$, $100\text{ K}\Omega$ and $1\text{ M}\Omega$, and with the resistors omitted to give infinite values), and on the control voltage, so that you will be able to choose values that will give particular centre frequencies and frequency ranges, and that you will know the corresponding values of the coefficient K_o , the constant of proportionality (in the linear regime) between the VCO's input voltage and output frequency.

7.5. Basic Closed-Loop Operation

You may wish to configure this experiment to run under computer control and measurement, and if so you should consider which of your available input oscillator options to use. You are still welcome to work in the old-fashioned way, with the signal generator box switches and dials to set frequencies.

Set up the complete phase-locked loop circuit, with oscillator and comparator circuits (perhaps with provision for injecting a square-wave signal from the external oscillator via a buffer so that the signal level is compatible with CMOS). Choose voltage-controlled oscillator external components for approximately a factor-of-two frequency range which lies within your range of input frequencies; for example, 50 kHz to 100 kHz range if the input signals are to be varied between a few tens of kHz and a few hundreds of kHz.

Close the feedback loop, trying both the type-1 comparator and the type-2 comparator. It may be useful to provide a variable resistor to attenuate the comparator output. Investigate experimentally the stability, lock range and capture characteristics. Consider also the variation with input frequency of the phase relationship between the input and output signals. Is it possible to lock onto signals that are harmonically related to the oscillator frequency?

With the current locked to an input signal, look at the waveforms on the capacitor of the voltage-controlled oscillator. Vary the input frequency. Are the waveforms consistent with the assumptions about circuit operation? How do the waveforms relate to the tracking range?

8. Phase-Locked Loop Integrated Circuit Applications

Now we will explore some simple experimental implementations of the 4046 chip. Again, the number of experiments completed is of far less importance than the quality of the investigation. Where there are two (or more) ways to approach the experiment, you need only to attempt ONE.

8.1. Frequency Multiplier

It is often useful to generate a waveform at a multiple of the frequency of a given waveform. For example, to remove the appearance of 50 Hz ripple on a cathode-ray tube display, the line and frame frequencies can be locked to the 50 Hz mains supply. Whilst this is now obsolete as an application for video displays, it remains useful in instruments such as scanning electron microscopes where alternating fields can deflect the electron beam and degrade the image. Many more applications for frequency synthesis exist in computer architecture and consumer electronics.

To implement frequency multiplication with a phase-locked loop, incorporate a divider in the feedback loop, Figure 28, so that the comparator causes locking to the corresponding sub-multiple of the frequency of the local oscillator. Either of the 4020-type counters may be used.

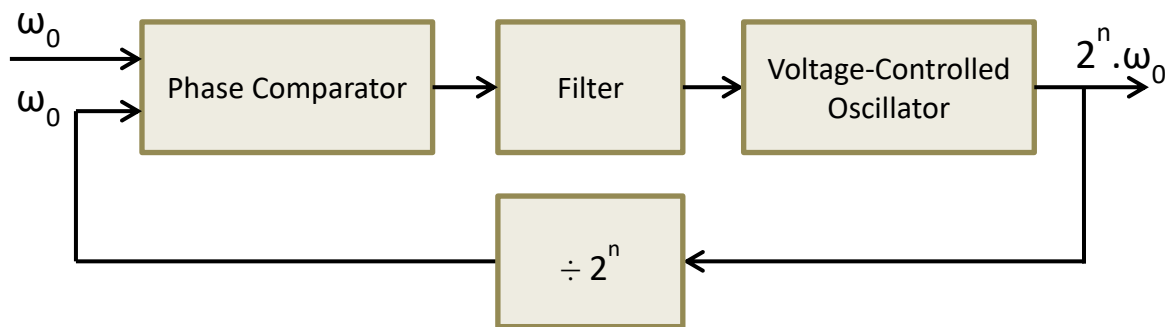


Figure 28 - Use of divider circuit to provide frequency multiplication.

Use the technique to generate a nominal 204.8 kHz signal from a nominal 50 Hz signal (from a signal generator, not the mains). Establish suitable oscillator and filter conditions for this application.

Consider the use of each of the phase comparators available in the 4046 circuit. Consider any unwanted effects that the comparator may introduce, and assess which is better for this application.

8.2. Transient Response and Frequency Modulation

The objective is to use the phase-locked loop to recover a signal that has been frequency-modulated onto a carrier, both to demonstrate the application and to act as a means for observing some transient characteristics for the circuit.

Once again, there are different ways of approaching this experiment, in terms of how to generate a frequency-modulated signal to demodulate. However, bear in mind that the experiment in the last paragraph of this sub-section is only appropriate to a 'breadboard-only' setup, which is described as follows:

It is convenient to use two 4046 integrated circuits, with the voltage controlled oscillator of one used as the source of the frequency-modulated carrier signal, and the other providing a complete phase-locked loop to act as a demodulator. The arrangement is shown schematically in Figure 29; the output should be taken from the source-follower output of the voltage-controlled oscillator, and may require some filtering.

To permit clear display of the signals, it is desirable to arrange both carrier and modulation to repeat in such a way that the composite waveform is itself periodic, for input to the PicoScope.

This requires that the signals are generated from a common source; a divider circuit may be used so that the modulation is a rectangular wave and both the higher-frequency and lower-frequency parts of the cycle contain the same number of cycles in the carrier frequency.

Optimise the filter for the recovery of modulation waveforms. It is suggested that the carrier frequency should be about 200 kHz, modulated by about $\pm 5\%$ (you may run slower if your setup dictates). Try different frequency-division ratios between the carrier and the modulated waveform to determine the highest frequency that can be used.

The modulation frequency is a square wave; hence the modulator introduces an abrupt change to the frequency of the modulated carrier, which the detector phase-locked loop must attempt to follow. This provides an opportunity to observe the form of the transient response, to an abrupt frequency change, for both type-1 and type-2 comparators. A useful trick here is to add the input and output signals using the 'Tools -> Math Channels' function of the PicoScope software or enable the 'A+B' trace on the LabVIEW PicoScope VI, and to consider the form of the resultant composite signal; this removes any ambiguity and any possibility of error in the relative phase of the two signals. Taking an external trigger signal from the divider output is useful.

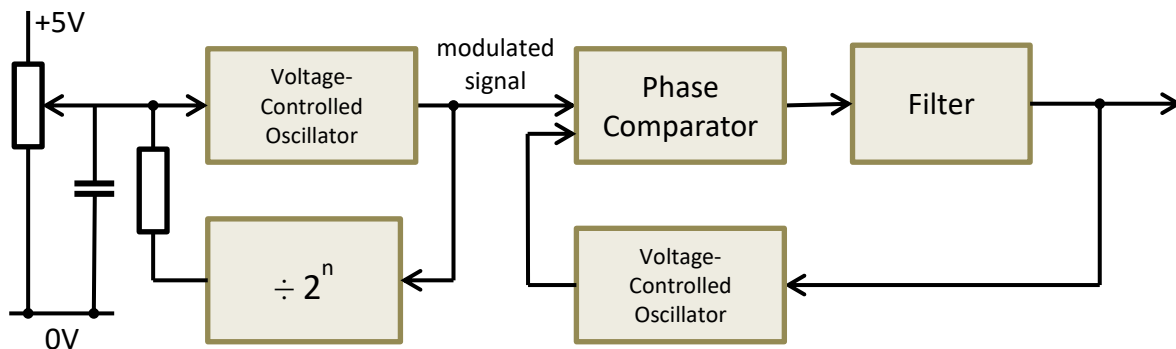


Figure 29 - Test circuit for transient response and for frequency modulation and demodulation.

In a real frequency-modulation system, the modulation signal has a defined bandwidth, narrow by comparison with the carrier frequency; the demodulator is not subjected to an abrupt change of input frequency. This situation can be simulated by adding a capacitor, as in Figure 29, to form an R-C filter at the input of the modulator voltage-controlled oscillator, whilst keeping the remainder of the circuit constant. Investigate the behaviour in this case, with a range of filter time constants.

8.3. Clock Regeneration

Another application for phase-locked loops is for the regeneration of clock signal at the receiving end of a data link in which pulses, synchronised to the transmitting-side clock, are sent down a single data channel. The regenerated receiving-end clock signal must be in phase with the single pulses that are received; however, these received pulses are intermittent because they represent data, and the received pulses may be distorted and noise may be present (Figure 30).

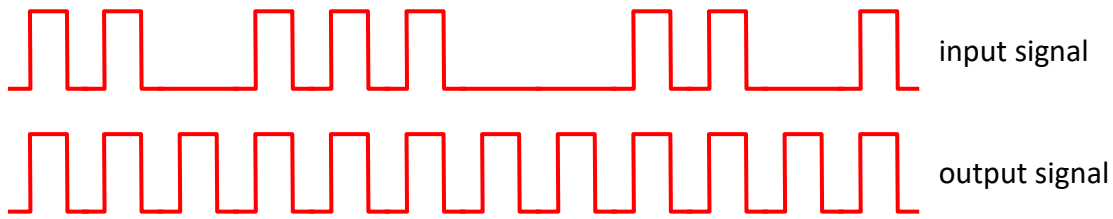


Figure 30 - Pulse train with missing pulses, regenerated to give a complete pulse train.

8.3.1. Characterising Clock Regeneration

You have a choice of two ways to approach this experiment, and should attempt only one of them. The first is the breadboard-based method of past iterations of this practical class, and is perfectly valid.

For both techniques, you should optimise the filter of the phase-locked loop for the recovery of the 'clock' waveform. The characteristics can also be modified by adjustment of the PLL's oscillator, both in respect of its free-running frequency and its adjustment range. Examine the dependence of the error signal immediately after the pulse burst starts on the difference between the carrier frequency and the free-running frequency, and on the choice of the ratio between the oscillator resistors R_1 and R_2 . As for the frequency-modulation trial, consider the results also for information about the transient characteristics of the circuit, and investigate the effects of changing the filter network.

Method 1

A convenient way of simulating a modulated pulse is to use a train of bursts of pulses, with a number of pulses followed by an equal interval in which the pulses are suppressed. The previous circuit can be modified for this purpose, with the output of the divider used to gate the 'carrier' signal, Figure 31. The carrier frequency is set by adjusting the d.c. input to the voltage-controlled oscillator.

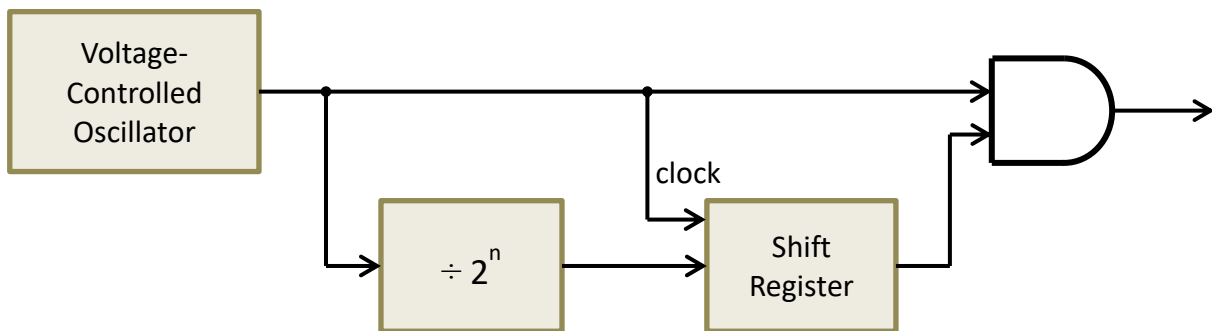


Figure 31 - Circuit for generating a train of bursts of pulses.

To facilitate viewing the waveforms, the modulation waveform can be delayed by a number of cycles using a shift register, enabling the non-delayed waveform to be used for triggering the PicoScope.

Method 2

Whilst providing less of an insight into mixed digital/analogue electronics, a computer-controlled experiment may be extremely powerful, allowing a large data set to be obtained. Combine a modified version of the LabVIEW control VI with the Arduino UNO and use it to provide a sensible variety of waveforms. As for Method 1, you should prepare a 'gap' in your pulses of a certain duration, though here you may prefer to observe how the detector PLL circuit behaves when a long pulse chain gives way to a long absence of pulses (*i.e.* repetition not necessary). The computer-controlled technique gives you the possibility of varying some parameters as you see fit.

8.3.2. *Recovering and Decoding a Real Signal*

In the Lab you will find an Arduino UNO which has been programmed to produce a message in digital form, repeated infinitely, its output coming from pin 12. There The format of the signal generated is sixteen 'zeroes', meaning 'absent' pulses, followed by sixteen 'ones' (5 V pulses), to signify the beginning of the code. Following that, the message is encoded in ASCII-format binary (in which code, for example, 'A' to 'Z' is represented by 65 to 90 decimal, or 01000001 to 01011010 binary). Use the clock regeneration technique to generate a clock signal appropriate to the data stream, recording both together. Comment on whether and/or how the waveform can be converted to binary and decoded, and, if you wish, do so.

References

1. **pico Technology.** PicoScope 2000 Series website. [Online]
<https://www.picotech.com/oscilloscope/2000/picoscope-2000-overview>.
2. **Arduino AG.** Arduino UNO Specification. [Online]
<https://www.arduino.cc/en/Main/arduinoBoardUno/>.
3. **Atmel Corporation.** ATmega328P Specifications (.pdf). [Online]
http://www.atmel.com/Images/Atmel-42735-8-bit-AVR-Microcontroller-ATmega328-328P_Datasheet.pdf.
4. **Arduino AG.** Arduino Reference Website. [Online] <https://www.arduino.cc/en/>.
5. **National Instruments.** NI Website. [Online] <http://www.ni.com/>.
6. **NXP.** Manufacturer's Data Sheets. [Online] <http://www.nxp.com/>.
7. **Cleaver, J R A and Weaver, D J.** Microelectronics Research Centre, Cavendish Laboratory, unpublished.

Selected Further Reading

- A. B. Grebene,** 'Bipolar and MOS analog integrated circuit design', Wiley (2003)
- P. Horowitz and W. Hill,** 'The art of electronics', Cambridge University Press (1989)
- S. M. Sze,** 'Semiconductor devices: physics and technology', Wiley (2002)

Appendices

A1. Experimental Apparatus

A1.1. The Prototyping Board

Circuits are assembled on an RS 489-100 prototyping board, Figure 32(a). Power is available from a +5 V/0 V paired terminal and a variable-voltage three-terminal supply; the two 0 V terminals are independent and float relative to each other unless suitably connected. Circuits are constructed by pushing components and wiring into the 'breadboard', which has sockets which are internally connected in horizontal banks of five, as shown in Figure 32(b). Power is provided to the breadboard by wiring the output terminals to the four supply rails, which run vertically as shown. Figure 32(c) shows the appropriate way to mount an integrated circuit across the central division to avoid short-circuiting opposite pins. Take care to identify which end is which; there is a notch (or occasionally a dot) at one end of every integrated circuit package to define orientation when using the pinout diagrams in Appendix A3.

Wear and tear on the breadboards means that occasionally there may be defects such as loose connections; similarly, integrated circuits do occasionally 'blow'. Bear these in mind, but remember that circuit failure is *almost always* the result of a wiring mistake.

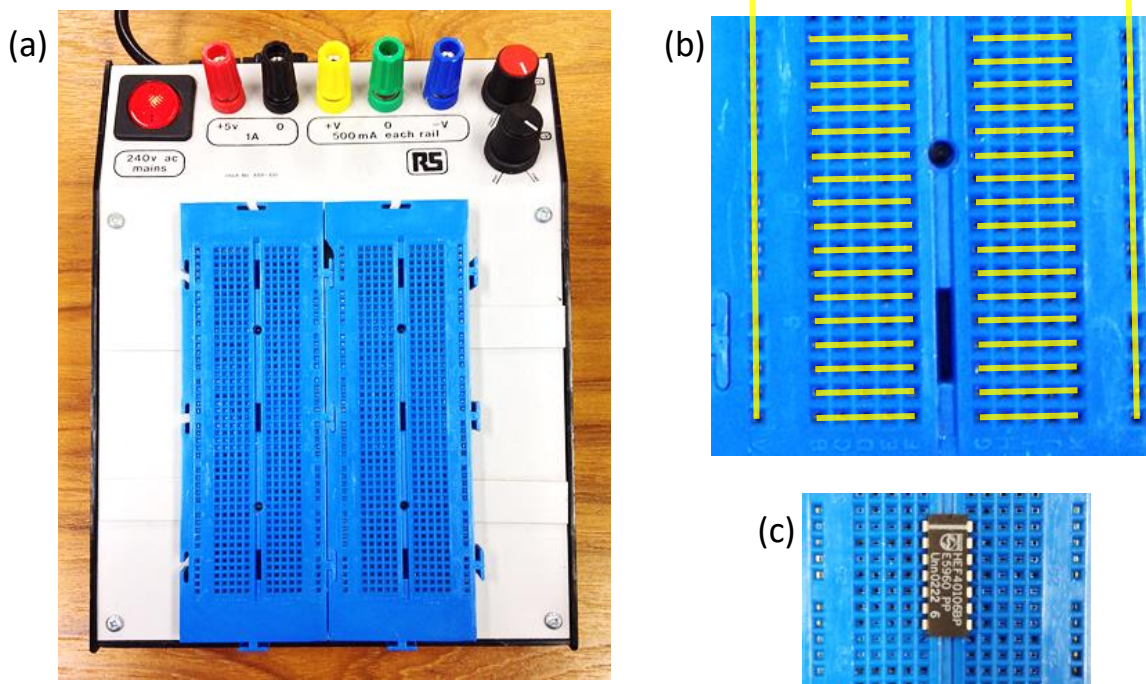


Figure 32 - RS prototyping board; (a) overview, (b) electrical linkage of pinholes with power lines on left and right, (c) correct positioning of integrated circuit across central gap.

A2. The Phase-Locked Loop as a Control System

A2.1. Linear, Small-Signal Analysis for Stability

A primary characteristic of any feedback system is its stability: that is, whether it provides constant output (for the phase-locked loop, a constant-frequency output) when the input conditions are held constant, or whether its output oscillates (here, the output frequency oscillates, so that the output signal is frequency-modulated).

We can consider the requirements for stable operation of the phase-locked loop by determining its response to small perturbations, whilst ignoring the process by which the circuit initially locks on the incoming signal. The linear, small-signal analysis of the system stability treats variations about the condition when the input frequency is equal to the free-running frequency of the oscillator. It is assumed that the comparator output voltage is linearly dependent on the phase difference, and that this voltage does not vary significantly over any one cycle of the voltage-controlled oscillator.

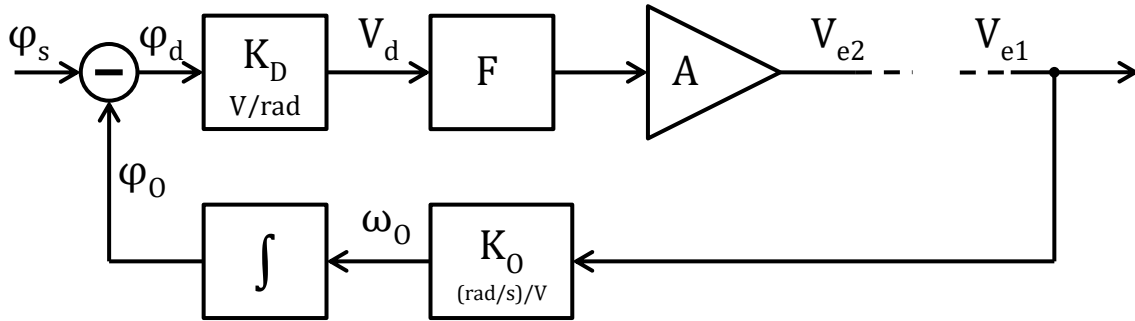


Figure 33 - Transfer function terms - circuit with feedback loop opened at V_e .

The feedback circuit is broken up into a sequence of 'transfer function' terms as shown in Figure 33. A standard way of assessing the stability of a feedback circuit is to consider its open-loop characteristics, and to predict from them what the characteristics would be with the loop closed. This is considered in more detail in A2.3, but the principle can be illustrated simply: suppose that the circuit is split at the output of the error amplifier, so that the amplifier output is separated from the input to the voltage-controlled oscillator. Suppose that a variable-frequency alternating signal is injected at the point at which the circuit is split (as the input to the voltage-controlled oscillator), and that for some injected-signal frequency the signal that returns to the other side of the split is in phase with the injected signal. If the magnitude and phase of the returning signal are the same as those of the injected signal, then the split could be reunited and the injected test signal removed and nothing would change - steady oscillations would persist. If the returning signal was in phase, but larger, then the signal would oscillate but its amplitude would grow exponentially; if smaller, its amplitude would decay exponentially. Thus we split the circuit at the error signal V_e to inject a signal V_{e1} into the voltage-controlled oscillator, and to investigate the signal V_{e2} (whilst keeping the input signal ω_s ($=d\phi_s/dt$) constant).

Assume that	$V_{e1} = V_{e10} + \Delta V_{e1}$
output frequency	$\omega_o = K_O (V_{e10} + \Delta V_{e1})$
output phase	$d\phi_o/dt = K_O (V_{e10} + \Delta V_{e1})$
but	$V_{e2} = K_D A F (\phi_s - \phi_o)$

with V_{e2} the sum of static and time-varying terms

$$V_{e2} = V_{e20} + \Delta V_{e2}$$

$$\begin{aligned} d/dt(V_{e2}) &= K_D A F (d\phi_s/dt - d\phi_o/dt) \\ &= K_D A F (d\phi_s/dt - K_O V_{e10} - K_O \Delta V_{e1}) \end{aligned}$$

Assuming the constant input signal and the unperturbed oscillator signals to be in phase, so that

$$\omega_s = \frac{d\phi_s}{dt} = K_O V_{e10}$$

then with complex sinusoidal variations at the modulation frequency ω_m ,

$$\begin{aligned} \Delta V_{e1} &= \Delta V_{e10} e^{j\omega_m t} \\ \text{and } \Delta V_{e2} &= \Delta V_{e20} e^{j\omega_m t}, \quad \Delta V_{e1}, \Delta V_{e2} \text{ complex} \\ \text{then } j\omega_m \Delta V_{e20} &= -K_D A F K_O \Delta V_{e10} \end{aligned}$$

In the basic case that the amplifier has flat response and negligible phase shift, and there is no filter,

$$\frac{\Delta V_{e20}}{\Delta V_{e10}} = \left(\frac{-1}{j\omega} \right) K_O K_D A$$

The phase-shift at the modulation frequency ω_m is always $\pi/2$, giving the Bode diagram, Figure 34. Therefore the system is inherently stable, since it would be necessary for the phase-shift to reach π for the system to oscillate (there is a further phase-shift of π due to inversion at the amplifier, since the system is designed to provide negative feedback).

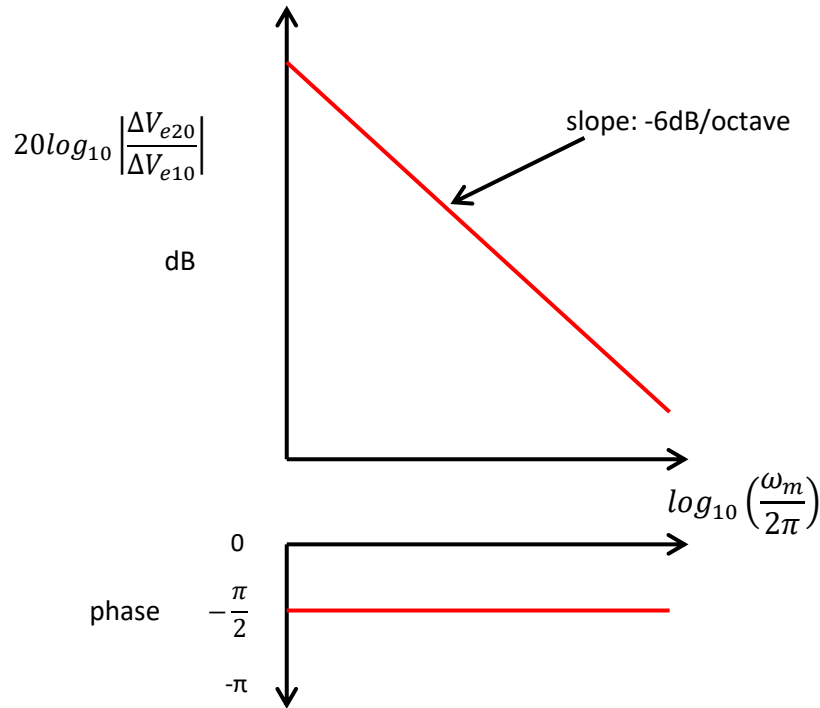


Figure 34 - Bode diagram for basic system without filter.

A2.2. Effects of Filter Circuits on Stability

The idealised basic system is inherently stable, but is very limited in its applications. Real comparator signals vary over a cycle of the signal frequency, and this variation may need to be smoothed out. It may be required that the voltage-controlled oscillator provides a steady output even if the signal is subject to noise or missing cycles (in clock signal regeneration, for instance) or if the frequency of the signal applied to the comparator is different from the frequency of the oscillator (as is a frequency multiplier).

In such cases, the obvious approach is to interpose an R-C integrator as the smoothing circuit, to filter the comparator signal before applying it to the voltage-controlled oscillator.

However, the effect of the integrator is to introduce further phase shift at the modulation frequency, so that as ω_m becomes large the added phase shift around the loop approaches π (and can exceed π if the other phase-shifts are present). Thus there is potential for ringing or even oscillation.

In summary, for the R-C integrator of Figure 18,

$$V_2/V_1 = (1+j\omega CR)^{-1}$$

To give Bode diagrams for the integrator alone and for the combination of the integrator with the remainder of the phase-locked loop as in Figure 35.

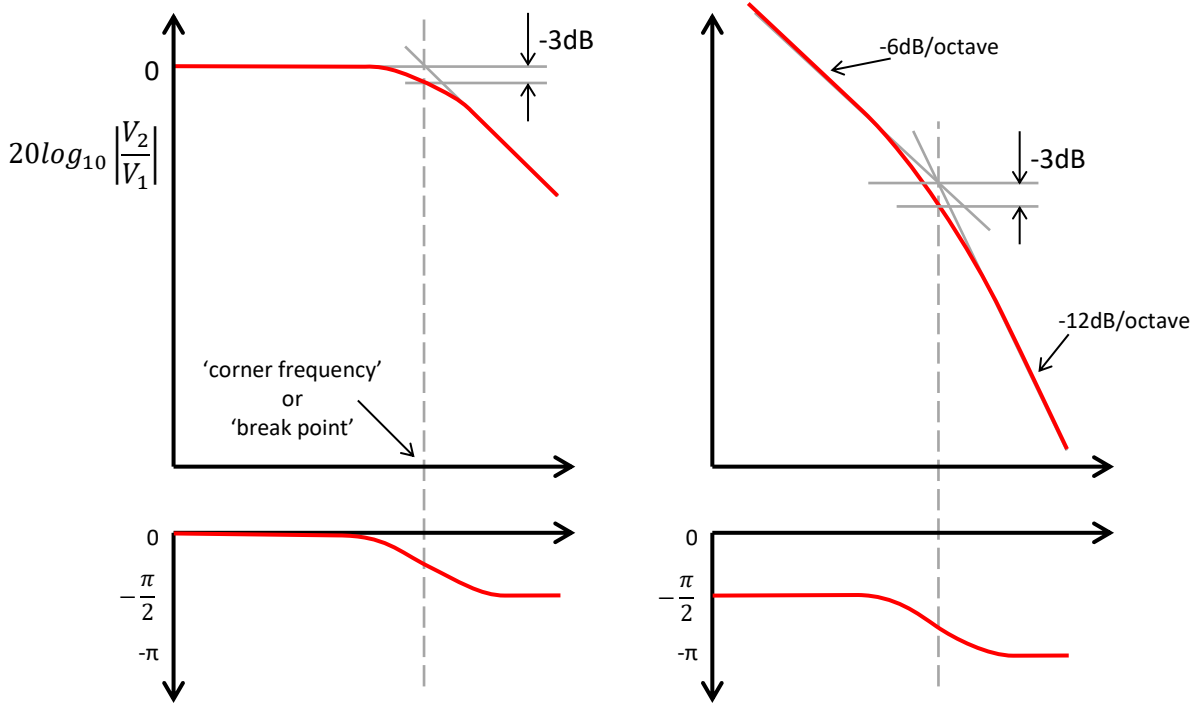


Figure 35 - Bode diagrams for the integrator alone, and for a complete system with an integrator.

A variant on the basic integrator is the lead-lag circuit of Figure 19, in which the phase-shift is zero at both low and high frequencies, and in which the phase-shift never reaches $\pi/2$.

In this case
$$\frac{V_2}{V_1} = \frac{1+j\omega C_2 R_2}{1+j\omega C_2 (R_1+R_2)}$$

or
$$\frac{V_2}{V_1} = \left(\frac{1 + \omega^2 C_2^2 R_2^2}{1 + \omega^2 C_2^2 (R_1+R_2)^2} \right)^{\frac{1}{2}} \left(\frac{\exp j\phi_1}{\exp j\phi_2} \right)$$

total phase-shift is
$$\tan^{-1}(\omega C_2 R_2) - \tan^{-1}(\omega C_2 (R_1 + R_2))$$

If the maximum phase-shift is evaluated, it will be seen to depend only on the resistor ratio, and so this ratio can be chosen to ensure an adequate phase margin for a complete system (irrespective of value of capacitor, which sets the frequency of maximum phase shift). The relevant Bode diagrams are shown in Figure 36.

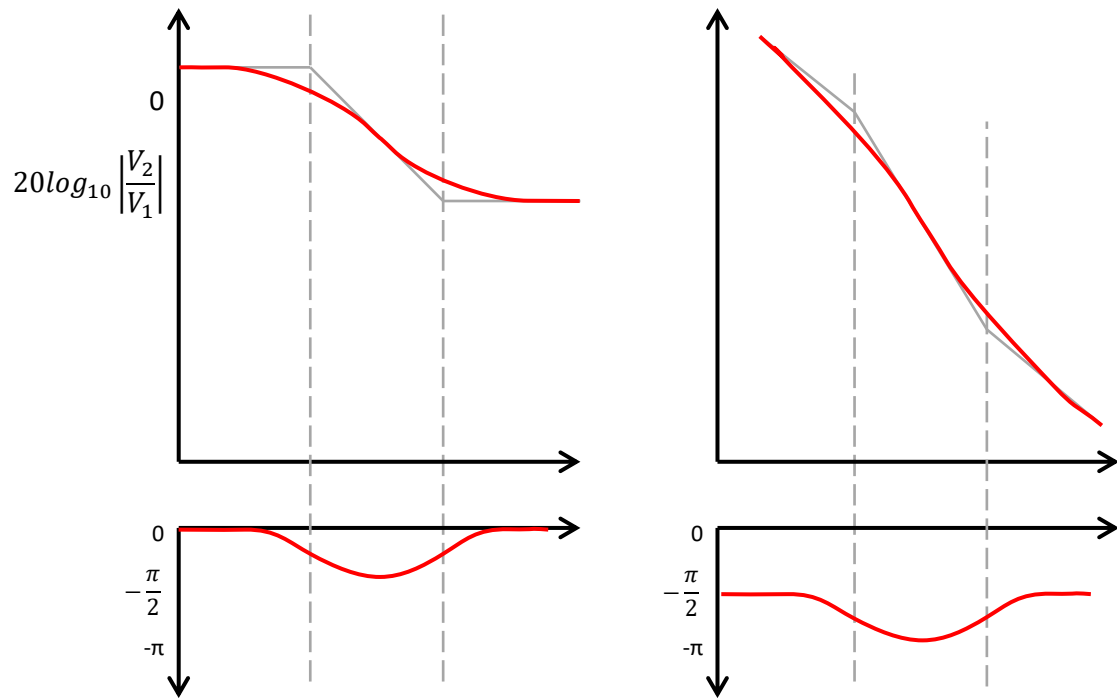


Figure 36 - Bode diagrams for the lead-lag circuit alone, and for a complete system.

A2.3. Gain, Margin, Phase Margin and Acceptable System Performance

The necessary condition for any feedback system, that it should not oscillate and should eventually settle to a static condition, is not sufficient for acceptable performance since it implies neither that the response to sinusoidal inputs will be satisfactory at all frequencies, nor that the transient response will be satisfactory. Examples of unsatisfactory behaviour are shown in Figure 37.

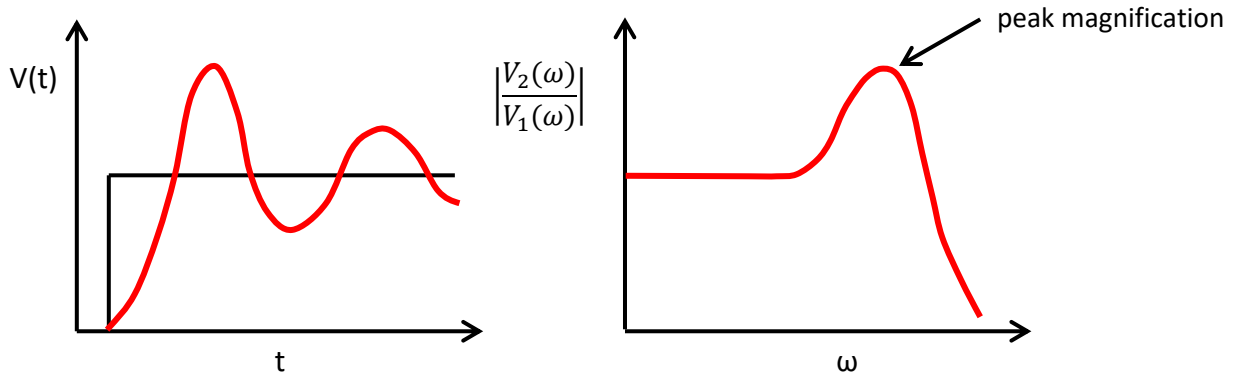


Figure 37 - Unsatisfactory response, shown in the time domain and in the frequency domain.

Consider a feedback system as in Figure 38(a), for which the open-loop characteristic V_2/ϵ is shown in Figure 38(b); this polar plot of the relationship between the 'error' signal and the output is known as a Nyquist plot. Then

$$V_1/\epsilon = V_2/\epsilon + 1$$

and the ratio between the input signal and the error signal on closed-loop is given by the open-loop plot with its origin shifted to $(-1,0)$; oscillation at steady amplitude corresponds to the open-loop locus passing

though $(-1,0)$, since then no input is required to sustain the closed-loop error and output signals, Lines of constant closed-loop magnification are plotted in Figure 38(c).

One criterion that can be applied is that (irrespective of phase shift) the closed-loop magnification should never be excessive, perhaps twice the low-frequency magnification. Then

$$2 |V_1/\epsilon| = |V_2/\epsilon|$$

This defines a region on the Nyquist plot, bounded by a constant-magnification locus, which the locus of the open-loop gain must not enter. Rather than using the constant-magnification locus directly, it is convenient to use criteria based on two specific points on the curve.

For **gain margin**: when the phase shift is π , the locus must not pass to the left of the marked point on the negative real axis. The amount by which the gain must fall short of unity (0 dB) is known as the gain margin, and is generally expressed in dB.

For **phase margin**: when the modulus of the open-loop gain is unity, the phase-shift must be less than π , by an amount known as the phase margin.

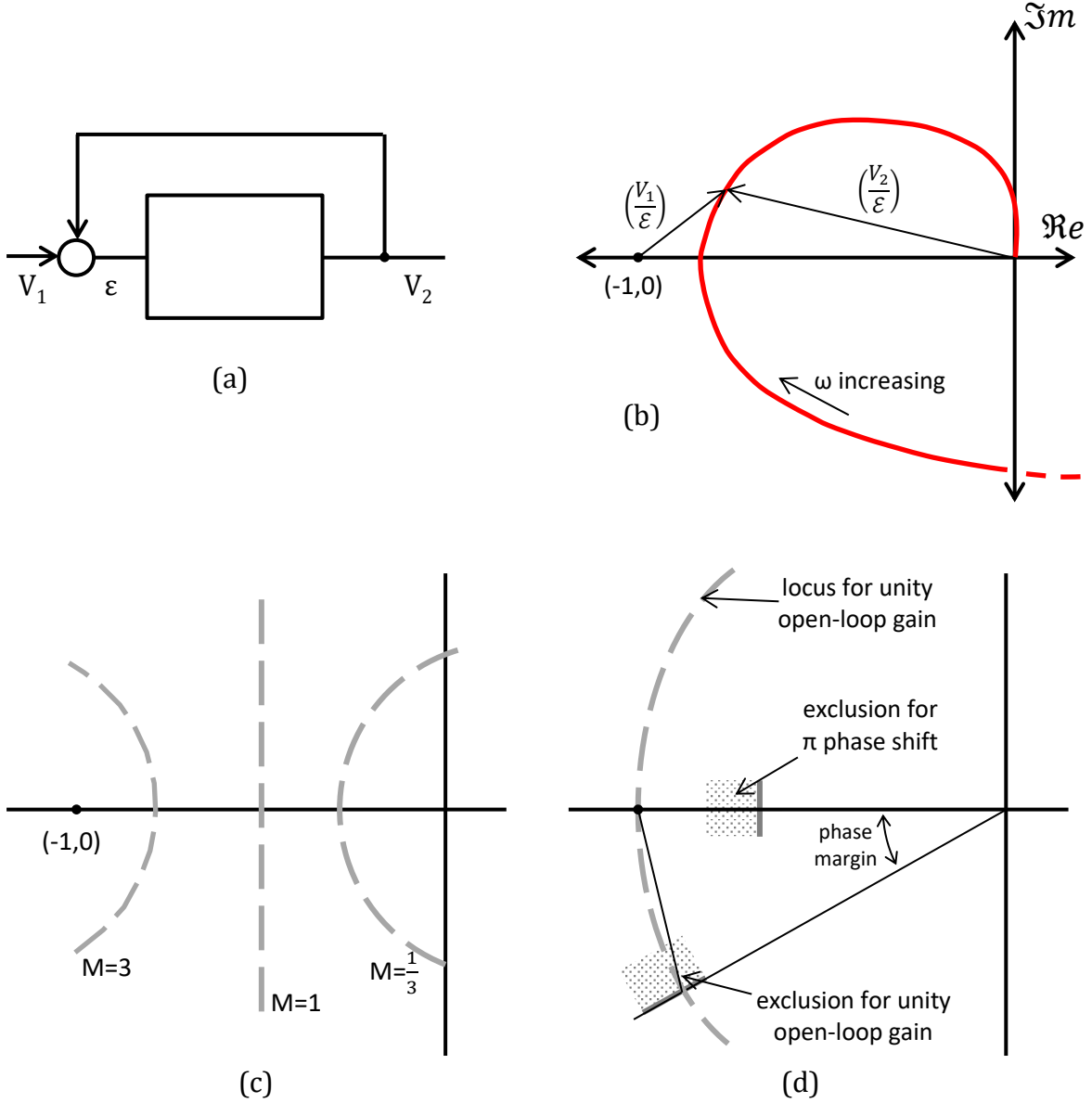


Figure 38 - Open-loop and closed-loop characteristics - the Nyquist plot.

A2.4. Bode Diagrams

The Bode diagram is a plot of the log of the modulus of the open-loop gain (expressed in dB) against the log of the frequency. It is supplemented by a plot of the phase shift against the log of the frequency.

The log-log form of the plot facilitates representation of system characteristics over a wide frequency range, and enables much of the plot to be approximated by straight lines. For instance, consider the simple R-C integrator, Figure 18, for which

$$(V_2/V_1) = (1+j\omega CR)^{-1}$$

In the low-frequency limit: $|V_2/V_1| \rightarrow 1$

In the high-frequency limit: $|V_2/V_1| \rightarrow 1/\omega CR$

The two straight lines intersect at $\omega = 1/CR$, where the value of $|V_2/V_1|$ is actually $1/\sqrt{2}$, -3 dB, and so the correction can be made and the characteristics sketched in.

Because the Bode diagram plots gain and phase separately against frequency, it is particularly suitable for use with gain-margin and phase-margin criteria. For example, the frequency at which the phase shift is π can be identified, and at that frequency the gain must be -(gain margin) dB; alternatively, at the frequency at which the gain has fallen to unity the phase shift must be (π -phase margin).

A3. Integrated-Circuit Connections

All the circuits used are in dual-in-line plastic packages.

All the HEF4000 series CMOS logic circuits can be operated off a +5 V supply.

All figures in this Section are reproduced from the manufacturer's data sheets [6].

A3.1. HEF4046B and 74HC4046 Phase-Locked Loops

The HEF4046B, on which this experiment is based, is currently unavailable from our suppliers. Please use the remaining HEF4046B chips, but if they run out, you will find 74HC4046, which are functionally the same apart from a third type of comparator (on pin 15, replacing the 'Zener' output on the old chip) which we will not be investigating here. Pinouts are shown in Figure 39.

NOTE: The 'inhibit' input must be grounded for the voltage-controlled oscillator to operate.

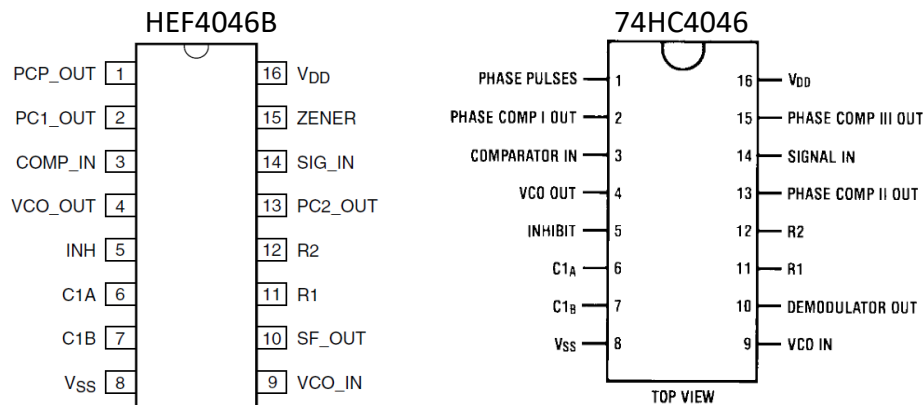


Figure 39 - The 4046 family of phase-locked loop integrated circuits.

A3.2. μ A741 Operational Amplifier

This circuit (Figure 40) is operated with symmetrical power-supply rails, here ± 15 V. There are many versions of the 741, and many alternatives (often with better performance) exist.

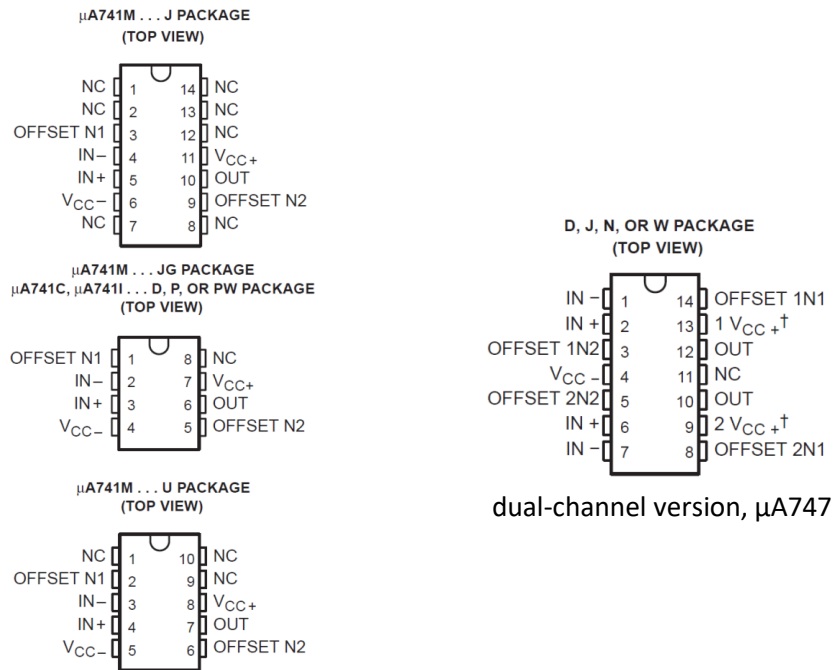


Figure 40 - μ A741 family of operational amplifiers.

A3.3. LM393A Dual Comparator

This circuit (Figure 41) can be operated with ± 15 -volt supplies, for which 'V+' becomes +15 V and 'GND' becomes -15 V. In this case, the output switches between +15 V and -15 V.

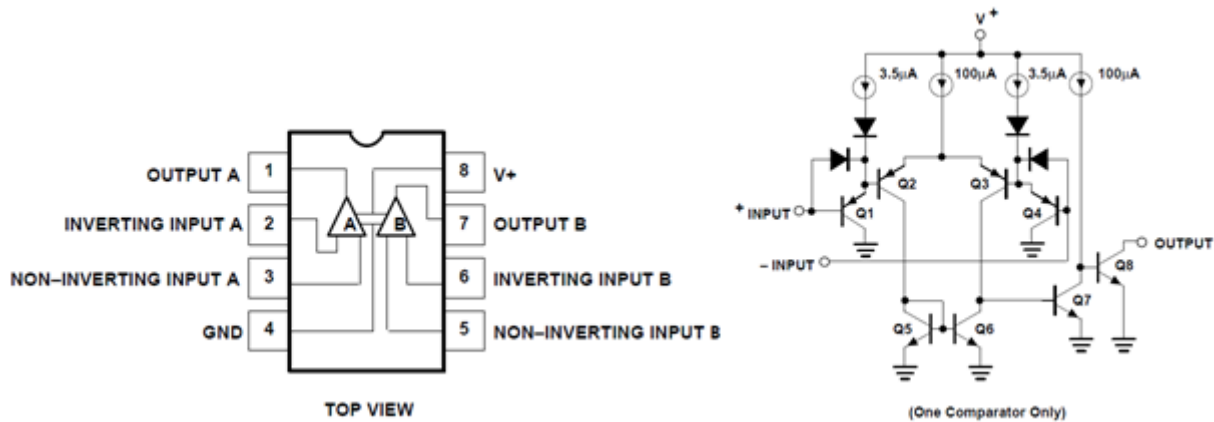


Figure 41 - LM393A Dual Comparator

Note that the comparator has an 'open-collector' output, with the collector of the output transistor connected directly to the output pin. In this arrangement, if the output transistor is turned ON, it can pull the output line down to 'ground' to give the 'logic 0' level, but when it is OFF there is no internal means for pulling the output up to the 'logic 1' level; this must be done externally, by connecting the output via a load resistor to a positive supply rail. The load resistor should be chosen so that the transistor current does not exceed 10 mA.

A3.4. HEF4015B Dual 4-Bit Static Shift Register

The circuit outputs (Figure 42) consist of successively-delayed equivalents to the input. The 'master reset' input MR should be held low.

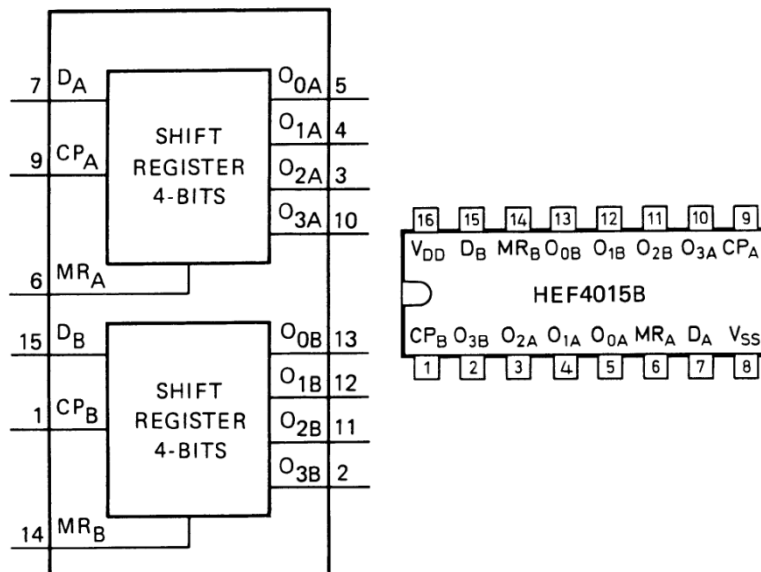


Figure 42 - HEF4015 shift register.

A3.5. HEF4020B and CD4020BE 14-Stage Binary Counters

The consists of a sequence of divide-by-two stages (Figure 43). The 'master reset' input MR (or 'RESET' on the CD4020B) should be held low. As with the HEF4046 phase-locked loop chip, the HEF4020 counter is no longer supplied, so you may need to use the (equivalent) CD4020BE (Figure 44) instead.

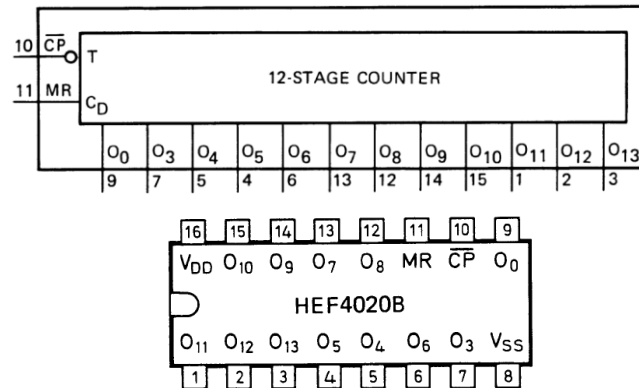


Figure 43 - HEF4020B binary counter.

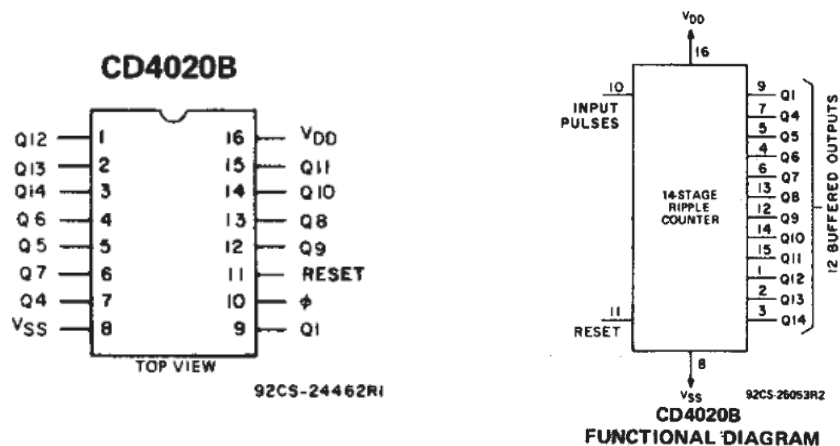


Figure 44 - CD4020B binary counter.

A3.6. HEF4081 B Quadruple 2-Input AND Gate

The HEF4081 B, Figure 45, is a seven-pin IC with four logic AND gates.

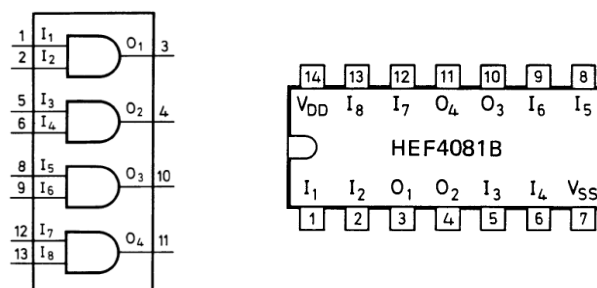


Figure 45 - HEF4081 B AND gate.

A4. Operational Amplifier Circuits

It is quite likely that you will not need to use a '741' operational amplifier, or 'op-amp', to modify a signal, but it can be very useful in some circumstances. These common circuits are provided for your reference, and without expectation that you will make use of this section.

Figure 46 shows some commonly-used op-amp configurations. The buffer circuit has unity gain, but its high input impedance makes it useful for minimising loading problems; the buffer supplies a voltage the same as that at its input even when load circuits draw high currents. The other circuits should not need describing, and their properties are straightforward to calculate.

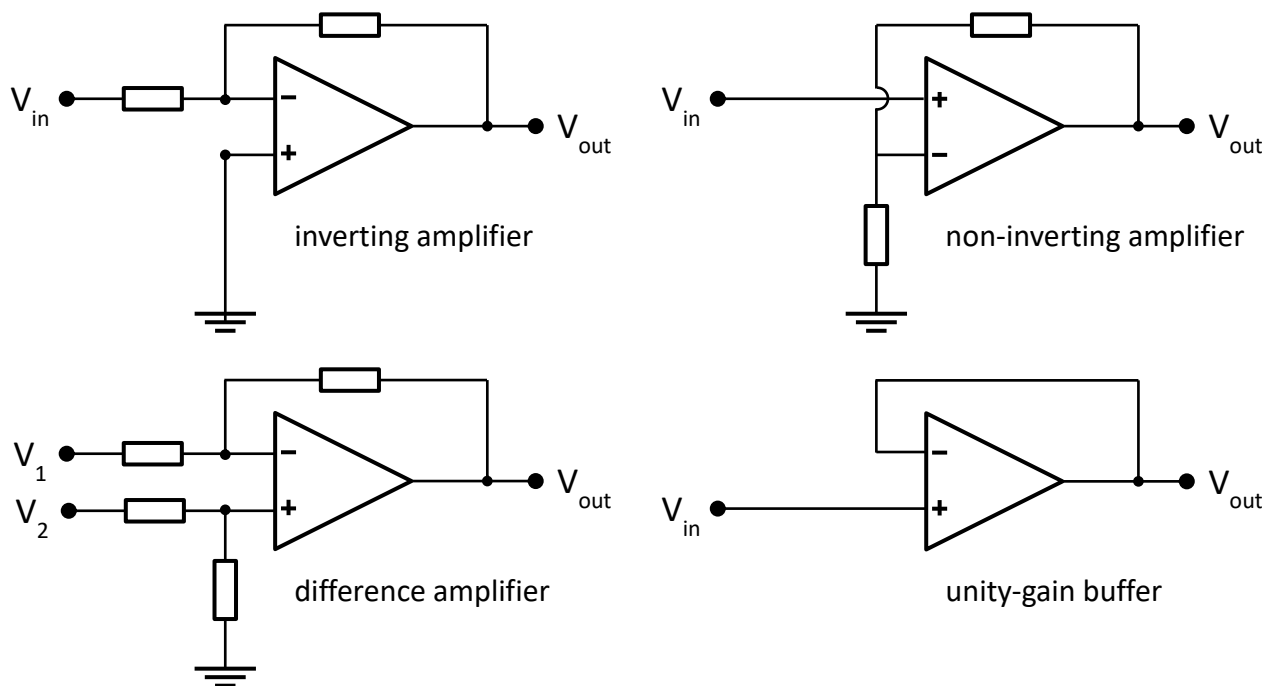


Figure 46 - Commonly-used configurations for operational amplifiers (note - supply voltages not shown).

A5. Physical Layout of the HEF4046B

In this section, some information is provided for understanding the physical construction of the 4046. It is included for interest only, having been part of an older version of this experiment. You are NOT expected to read or understand this section!

A5.1. Introduction

It is well worth acquiring an insight into how the HEF4046B chip works as well as what it does. Despite its great simplicity and large feature size compared with modern processors and memory chips, the HEF4046B still appears rather complex. However, once you have an idea what you're looking at, many features can be identified. The numbered input connections on the micrograph (Figure 26) can immediately be matched up with the corresponding chip package terminals (Figure 20), but some further knowledge of chip manufacture will be helpful.

If you have been working with 74HC4046 chips, consider them to be essentially the same as the HEF4046B.

A5.2. Example 1: Four-Input NOR Gate

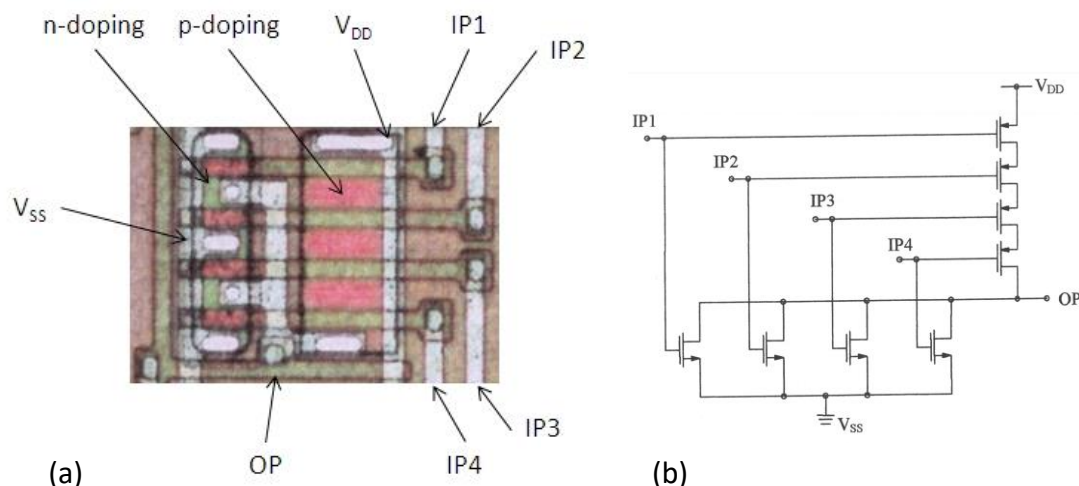


Figure 47 - (a) Micrograph of a four-input NOR gate from a HEF4046B and (b) its equivalent circuit.

As an example of one of the basic units of the HEF4046B, Figure 47 shows a four-input NOR gate (found in Figure 26 at centre-left), along with its circuit equivalent. Some correspondence between the two figures can be picked out immediately - the vertically-aligned silver-coloured tracks on the right of Figure 47(a) are metal interconnects feeding in the four input voltages, IP1 to IP4. These link to greenish horizontally-oriented tracks which play a dual role - like the metal tracks, they are simple electrical conductors, but they also act as the gate electrodes for transistors. They are made of implantation-doped polycrystalline silicon.

It should now be possible to pick out two banks of four transistors in both parts of the figure; the large pinkish rectangle in the centre is an area of p-type silicon, and belongs to the bank of transistors running vertically in the schematic of Figure 47(b). Notice that the four transistors connect in series, having metal contacts to the outside world only at the ends of the bank. By contrast, the other set of transistors, which run horizontally in the schematic and vertically (at the left) in the micrograph, are connected at each intermediate point.

One remaining issue is isolation. The micrograph has numerous apparent intersections where conductors cross, for example where the vertical V_{DD} line runs over the IP1-4 gates. In fact there is a layer of insulating material, probably SiO_2 or similar, between the silicon/polysilicon parts and the metal tracks, and the metal makes contact with the buried devices only through small holes etched in that insulator - you can see several of these holes in the micrograph, most obviously on the IP1-4 inputs. The insulator is probably grown by oxidising the chip in an oxygen furnace, or perhaps by one of several layer deposition techniques.

A5.3. Process Flow

A likely process flow, somewhat simplified, for the four-input NOR gate of Figure 47 is shown in Figure 48. Processes will be more complicated in the actual fabrication line, and in reality will differ significantly from this illustrative flow.

The process starts with a (probably lightly p-doped) silicon wafer (a), perhaps having a buried SiO_2 insulator layer (not shown) which helps with good electrical isolation and performance. The transistor areas are defined by masking, perhaps with SiN , and the wafer is given a prolonged oxidation in a furnace under oxygen flow before the mask is etched off, leaving the majority of the wafer's surface covered by highly insulating SiO_2 (b). A fresh layer of silicon (which will change from amorphous to poly-crystalline during later thermal treatments) is deposited across the whole chip by sputtering then etched away where it is not needed, defining transistor gates and some other interconnect tracks (c). High-voltage implantation of dopant ions will make the poly-crystalline silicon highly conductive, and is also used to define the transistors' drain and source contacts by doping the silicon (d). An insulating layer is formed across the wafer surface (e) by a further oxygen furnace step, or perhaps by spin-coating with a glass precursor and subsequent heating. Where contacts are needed to the buried structures, holes are etched in the insulator (f), before the final metallisation is performed (g), again by sputter-depositing metal across the whole wafer and etching away the unwanted material. A metal such as aluminium (saturated with a few percent silicon to prevent substrate dissolution) may be used in low-tech applications.

A key point to bear in mind is that the metal tracks only ever make electrical contact with the polysilicon tracks via the etched holes in the dielectric. The two layers cross each other in very many locations without making any electrical connection.

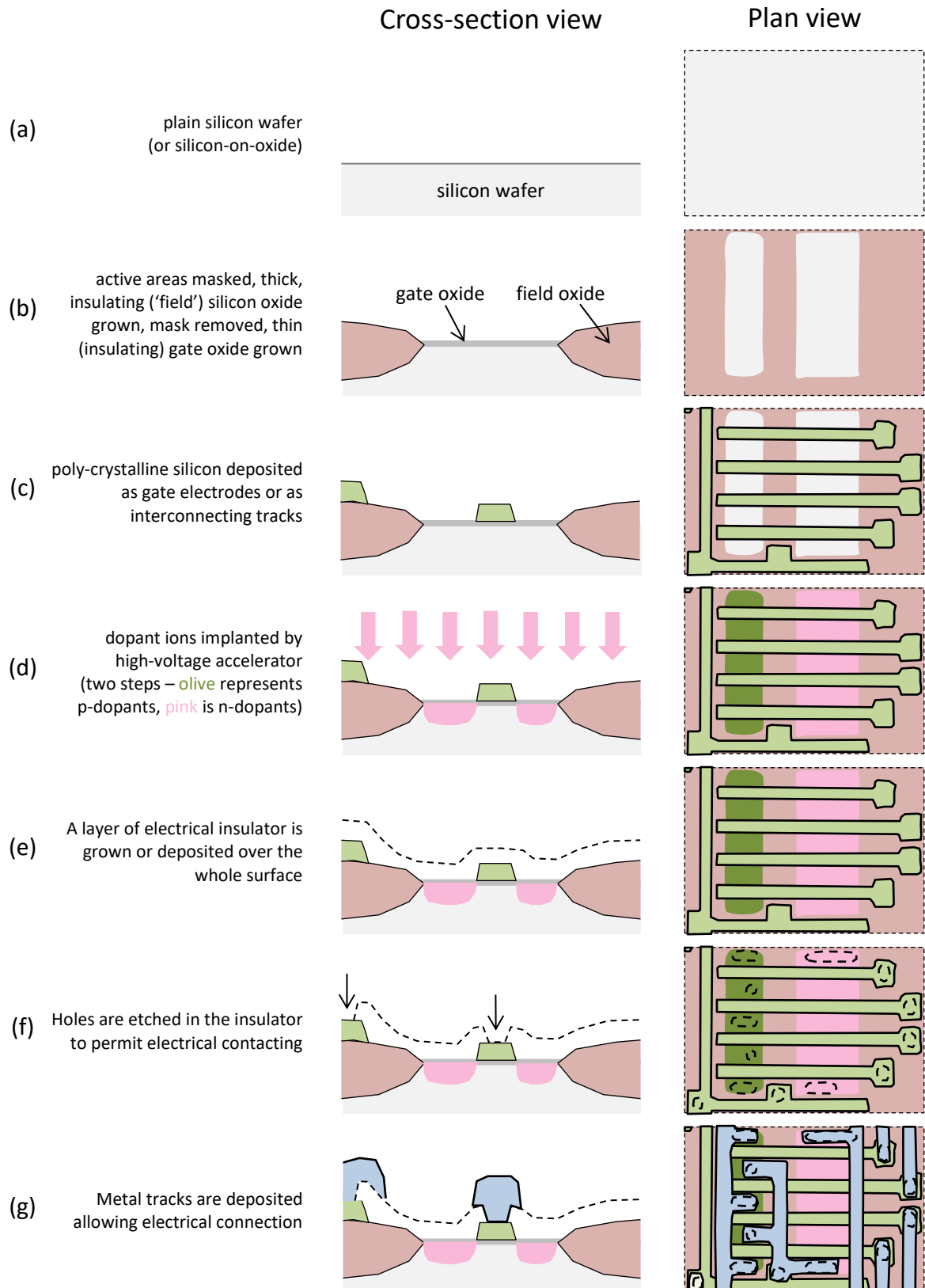


Figure 48 - Plausible simplified process flow for the HEF4046B chip, with the layout of the four-input NOR gate of Figure 47 as an example.

A5.4. Example 2: Large Channel-Width Transistor

The HEF4046B chip needs to control some fairly large currents (compared with those used for logic) for certain of its duties, and therefore it incorporates some high-current transistors. To accomplish a suitably high current flow, a design with a large channel width is incorporated. One such transistor is examined in Figure 49. By interdigitating the polysilicon gate (b) with metal drain/source contacts (d), the block is equivalent to a single transistor of width several times the physical width of the block. Note that the vertically-oriented metallic strip in (d) crosses the insulating dielectric and therefore is only in electrical contact with the 'source'-type rows of the device. Similarly, the metal strips of the 'drain' contacts either side of this spine are joined beneath it via the doped region in the substrate. Other transistors on the HEF4046B have different configurations, but similar principles in terms of interconnection.

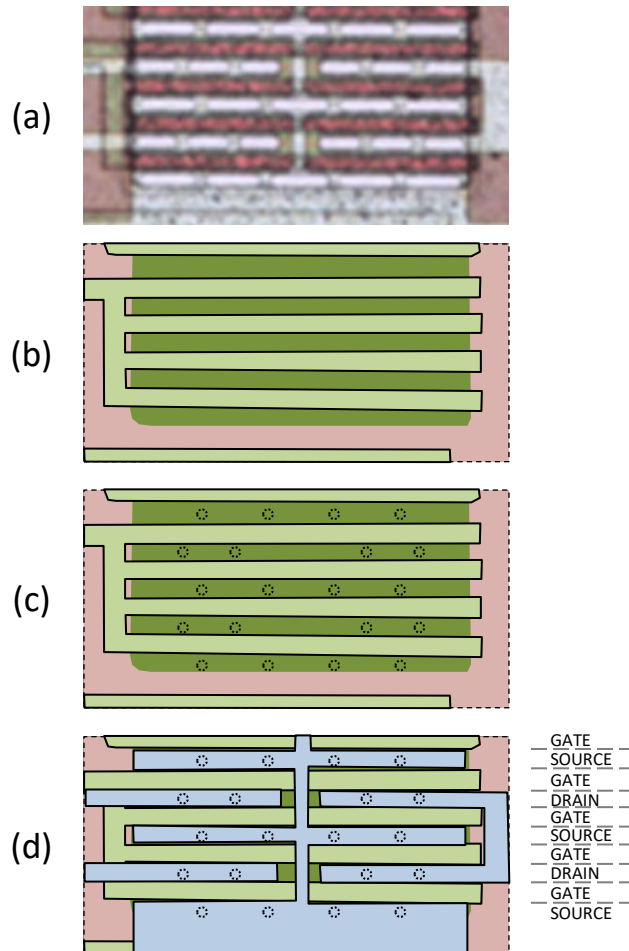


Figure 49 - Large-area transistor from HEF4046B chip showing (a) micrograph extract, (b) position of gates over active region, (c) electrical access holes etched through insulating dielectric, (d) metallisation, indicating the purpose of each horizontal strip.