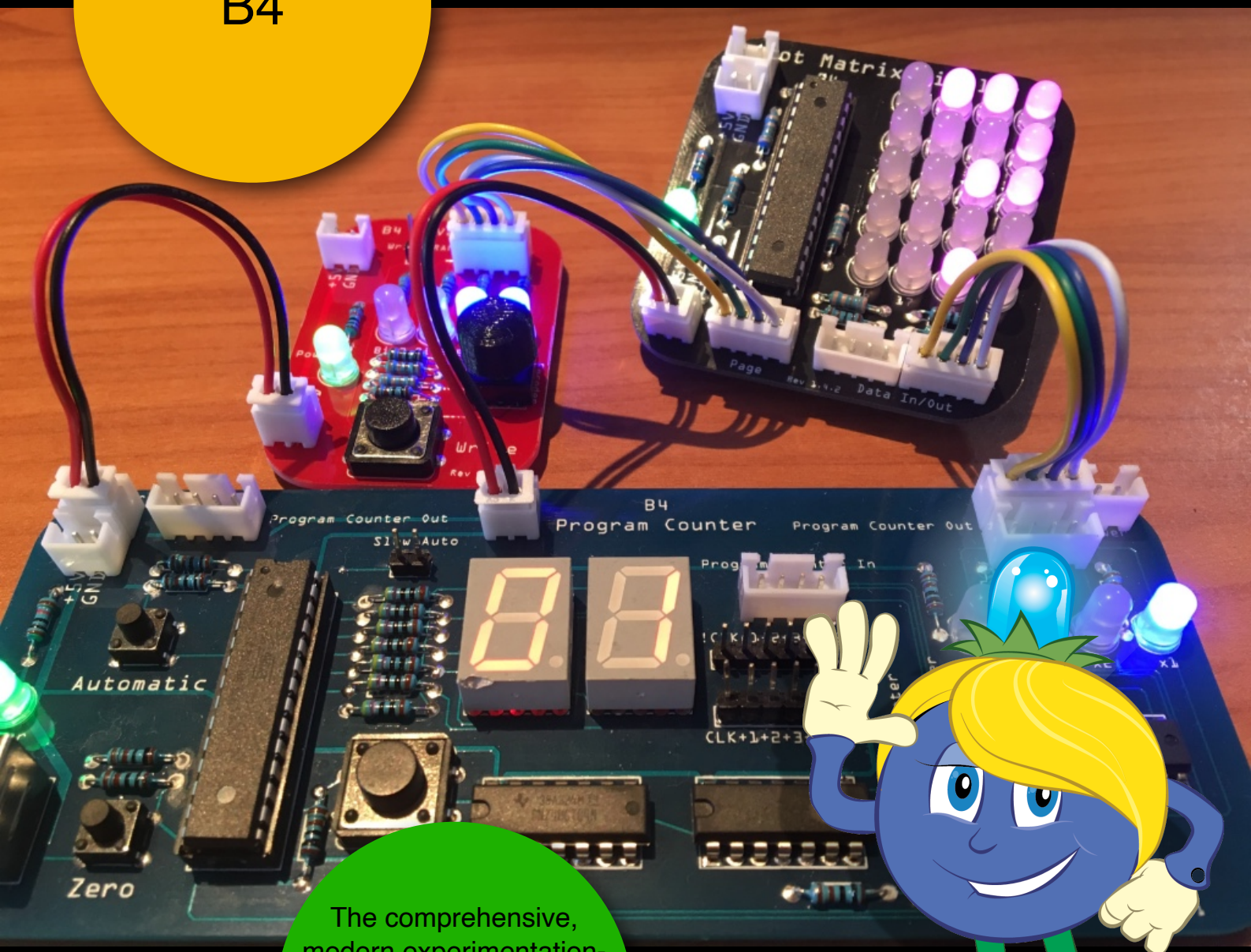
B4



The comprehensive, modern experimentation-based course unlocks the understanding of the visual world of Digital Technologies.

GRAPHICS EXTENSION KIT SPARK EDITION

0010
0000
0010
0110

DIGITAL TECHNOLOGIES INSTITUTE

“A designer knows he has achieved perfection not when there is nothing left to add, but when there is nothing left to take away.”

(Antoine de Saint Exupéry)

Table of Contents

WARNING:
CHOKING HAZARD - Small Parts
Not for children under 3 years.
PHOTOSENSITIVE EPILEPSY - Some of the missions produce light flashes that can potentially trigger seizures in people with photosensitive epilepsy

Safety instructions

The B4 operates on 5 Volts and only draws a few milliamperes. Nevertheless, it is an electrical device and should be handled as such. We recommend treating it with care and keeping it on a non-conductive, dry and level surface. Do not scratch the surface of the printed circuit boards with sharp or metallic instruments, as this might damage the wires.

Acknowledgements

We would like to thank Charles Petzold, the author of 'Code: The Hidden Language of Computer Hardware and Software', published in 1999. His book has both inspired and guided the design of the B4. We recommend it as additional reading material for students.

We would further like thank Henrik Maier from proconX for his guidance and feedback on the electrical engineering design, fabrication and component selection, which has been invaluable to transforming the B4 from a breadboard prototype to a robust design that can be used in the classroom.

Special thanks to Dr. Hayden White for his support and input which have been invaluable to get the B4 off the ground. His regular feedback on the development of the B4 has influenced many of the design decisions.

Esther and Michael Schulz have contributed to the design of the characters that are stored in the Dot Matrix Display.

Mrs. Sharon Singh and her year 8 students at St. John's Anglican College in Forest Lake, QLD, have provided great input on the design of the Dot Matrix Display

A big thank you is owed to the Arduino community. The B4's Dot Matrix Display module deploys an Atmega processor which runs an Arduino program. Keep up the great work !

Dr. Karsten Schulz, CEO, The Digital Technologies Institute.

Included Parts

1x Dot Matrix Display Module
1x Data RAM Module

2 x 4 Pin Wires
2 x 2 Pin Wires

1x Student Handbook

Not included
B4 Spark) Kit, sold separately. The missions in this handbook require parts from the B4 Spark Kit.

Power Consumption:
5V, 50mA, 250mW, DC.

This product complies with the Restriction of Hazardous Substances Directive and is lead free.

The illustrations in this handbook can slightly differ from the actual modules. However, the functionality is the same.

This handbook has been made with great care. Should you find errors or have ideas to improve it, please email us at enquiries@digital-technologies.institute.

Designed and manufactured in Australia

Welcome back, Parents and Teachers

The B4 Spark kit explored the fundamental operation of computers with core functions, such as data storage, addition and subtraction with underlying algorithmic design. This extension kit adds graphics output capabilities to the B4. The B4's Dot Matrix Display Module can output ASCII-style characters and symbols on a 4 by 5 LED matrix. In addition, students can program 16 of the LEDs separately and thus design their own graphics. This lesson plan follows the B4-style bottom-up motivational approach, in which computing concepts are introduced by need and motivated by context. Students gain an understanding of the practical necessity of computer graphics concepts and learn about them from a hardware and a software perspective. The result is a deeper and more natural understanding of Digital Technologies.

This kit has been designed with the new Australian Curriculum: Digital Technologies in mind.

Welcome back, Students

Congratulations on graduating from the B4 base course. By completing the missions from the previous course, you have gained a solid understanding of how a computer really works inside. With this new kit in front of you, we will expand the B4's capabilities towards producing graphical output, such as letters, numbers, smileys and so forth. You will also learn about pixels and animations.

Besides following this study guide, we again encourage you to conduct your own missions and try things that are not in this handbook. You never know what you might discover.

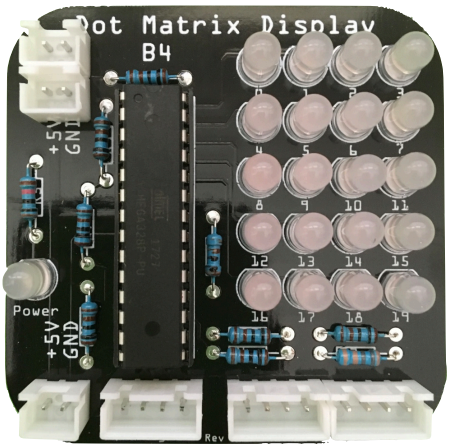
Now let's see what's in the box:

B4's Graphics Extension Kit Parts

The B4 Graphics Extension Kit consists of two modules which extend the B4 Spark kit that you already have. One module, the Dot Matrix Display, is completely new. The other module is a Data RAM modules, as some of our missions require the presence of two Data RAM modules.

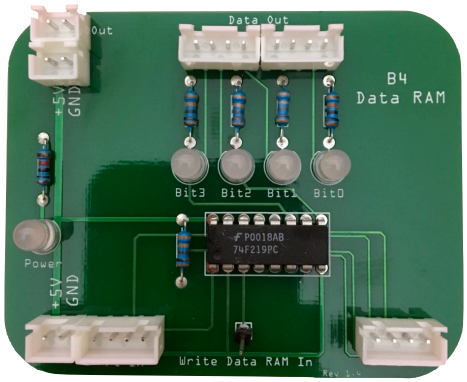
Generally, each module receives its input through the connectors at the lower end of the module and provides an output through the connectors at the top. All connectors are labeled with In or Out.

The modules are all labeled. Take them out of the box and inspect each of the modules as we describe them.



The **Dot Matrix Display** has 20 pixels, which are arranged in a 4x5 matrix. At the bottom of the module is a page input port and a data input port. With these we select which pattern we want the pixels to show, for example a letter, a number, a smiley, or something else. For a summary of all the patterns that the display can show, refer to Appendix A.

The **Random Access Memory (RAM)** looks a bit different, but is identical in function to the **Data RAM Module** from the B4 Spark Kit.



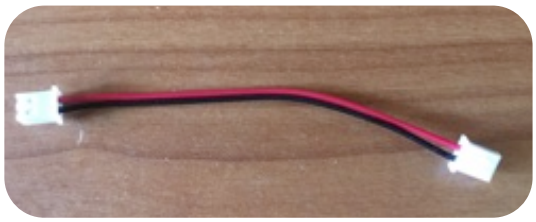
We now have a basic understanding of the modules of our B4 Graphics Extension Kit. Don't worry if you haven't understood everything yet. We will revisit each module in more depth during the following missions.

Wires and Connectors

In order to connect the B4 modules with the computer and with each other, you will use the following wires throughout the missions. The Spark kit already provides most of them, and this extension kit mainly adds some additional 2 pin and 4 pin wires. Still, we thought to re-introduce all the wires to re-familiarise yourself with them again. They are:

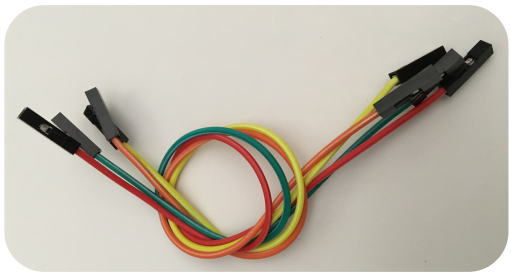
A **USB cable** to provide electricity from a power source to the B4's Program Counter module and from there, to all other modules connected to the Program Counter. You can connect the USB cable to a PC, Laptop, USB Hub, USB battery, or any other suitable 5V power source with a USB port.

2 pin **power wires** with black/white and red wires. They are part of the B4's power distribution system and transport electricity from module to module. Each module has one power input and 1-2 power outputs.



4 pin **data wires**. These transport 4 bit data and program counter signals from the output of one module to the input of another module.

1 pin **control wires**. They transport operation codes and instruct some of the modules of the B4 to do special things, such as storing data. The 1 pin wires come in many different colours. However, they all work the same and their colour has no influence on their function

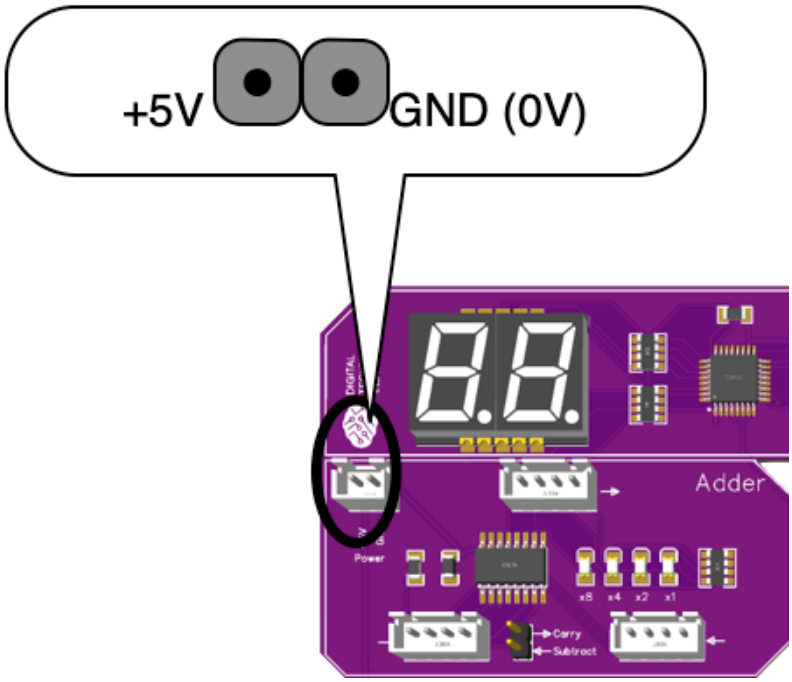


You will find corresponding connectors on the modules. The 2 and 4-pin connectors are directional, and the wires will easily click into them. Unless you apply excessive force, you should not be able to accidentally plug them in the wrong way.

A Word about Power

The B4 has an electric power distribution system. There are four white two-pin ports on the B4 Spark, and the Variables have two each.

+5V is on the left and GND (Ground, or 0V) is on the right. The power wires will automatically connect in the correct way, **but sometimes we will need to connect a single wire to either +5V or GND during some of the missions**. When asked to connect to +5V, just plug a single wire into the left pin of the power node. If asked to connect to GND, plug a single wire into the right pin of a power node.



Please look after me

The B4 is fairly robust and will last a long time with proper care. As long as you don't plug wires into connectors that are not designed to fit, and as long as you don't drop the modules, step on them, or use them as a doorstop, things should be just fine.
Always only plug the 2-pin wires into 2-pin connectors. The same applies to 4-pin wires and connectors. **Under no circumstances plug a 2-pin wire into a 4-pin connector**.

About Binary and Decimal Numbers

To be clear about the distinction of binary and decimal numbers, we add a capital 'B' to binary numbers. This way we can distinguish for example 11 (decimal eleven) from B11 (decimal 3), or 10 (decimal ten) from B10 (decimal 2).

Ok, that is enough preparation for now. We will collect more details as we work through the missions. Let's get started.

Missions

Overview
In this handbook, we have prepared several missions that will help you to explore computer graphics. You will learn about character output, pixels and drawing on the display to produce beautiful patterns and artwork.

We recommend that the missions be taken in sequence. But if you are already a computer genius, feel free to jump around. We should mention, that the B4 can do much more than what is written in this handbook. Feel free to explore and try out different things as you like.

	Title	Learning Objectives
1	Producing Graphics Output	Getting to know the Dot Matrix Display. Accessing letters, numbers and symbols.
2	Automating Graphics Output	Storing references to characters in the Data RAM. Executing a program to call them in sequence.
3	Do-Re-Mi	Extending our hardware setup to call any character from any page. ‘HELLO’ output example
4	Pixels	What are pixels? Free drawing on the Dot Matrix Display
5	Making Maths Artwork	Using the B4 to produce beautiful mathematical artwork simply by adding numbers and displaying their pixel value on the Dot Matrix Display.

Mission 1: Producing Graphics Output

Modules Required: Program Counter, 2x Variable, Dot Matrix Display

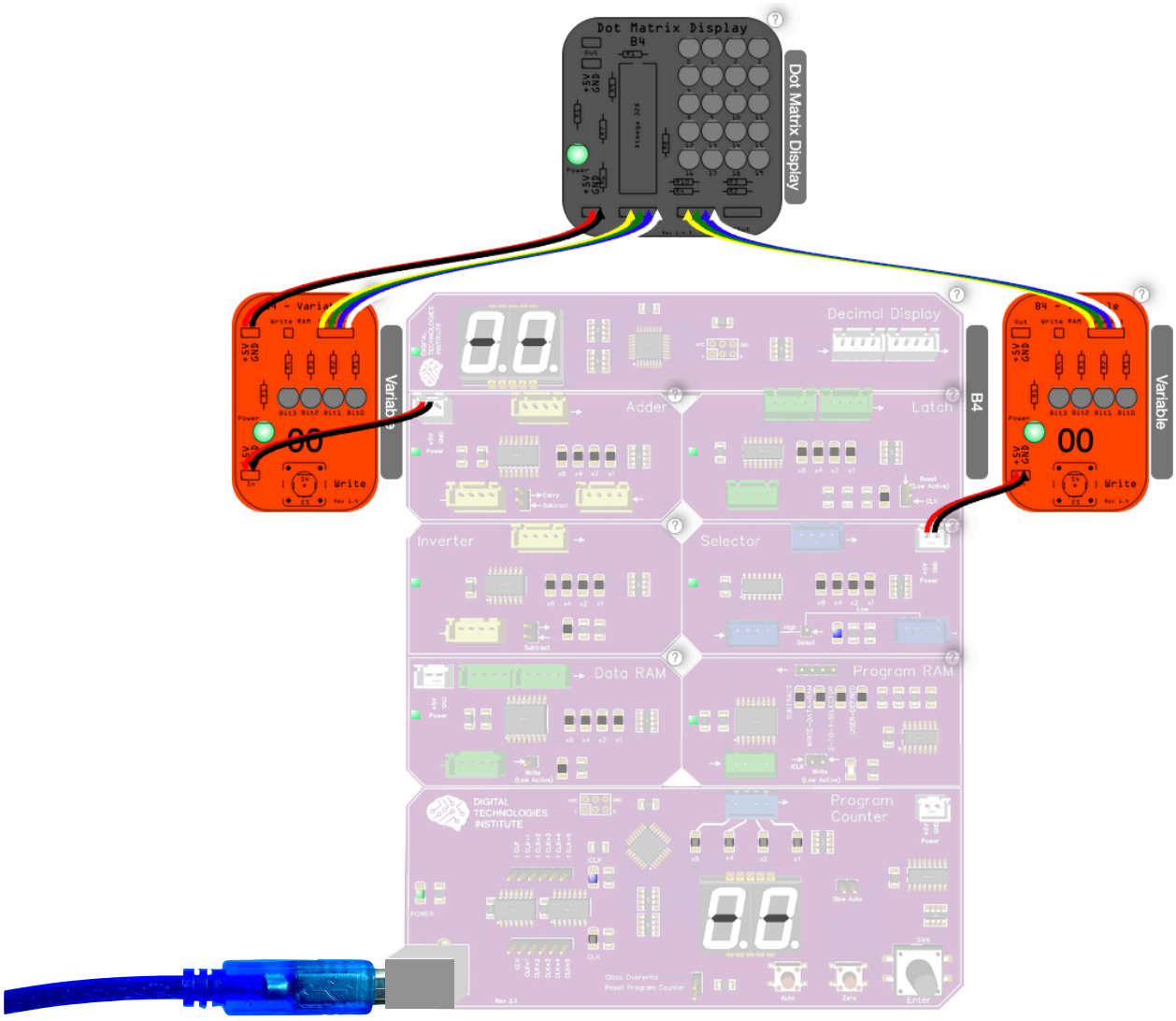
Objective
Our aim is to get to know the Dot Matrix Display. Since it has two inputs, called page and data, we provide binary data to these ports and observe what happens.

Build
1) Setup and Power A. Place the four modules on the desk in front of you as shown in the following diagram. B. Connect the two Variables to the Program Counter with power wires as shown. C. Use a power wire and connect the left Variable with the Dot Matrix Display. 2) Data A. Connect the 4 pin output of the left Variable to the Pageconnector of the Dot Matrix Display. B. Connect the 4 pin output of the right Variable to the Data In/Out port of the Dot Matrix Display. There are two Data In/Out ports on the Dot Matrix Display. You can use either. The second one is useful when we want to mirror the output to a second Display. This is not important at the moment. Just keep it in mind. The following diagram shows the setup of this mission.

Mission 1.1
1) Connect the Program Counter to a USB power source. 2) Set both Variables to B0000.

Observe 1.1
What is the output of the Dot Matrix Display?

Compare 1.1
The Display should now show the letter ‘A’.



Setup of Mission 1

Mission 1.2
Turn the right Variable to B0001.

Observe 1.2
What is the output of the Dot Matrix Display?

Compare 1.2
The Display should now show the letter ‘B’.

Mission 1.3
Turn the right Variable to B0010

Observe 1.3
What is the output of the Dot Matrix Display?

Compare 1.3
The Display should now show the letter ‘C’.

Mission 1.4
Continue incrementing the right Variable until all LEDs are on, which is B1111.

Observe 1.4
What is the output of the Dot Matrix Display?

Compare 1.4
The Display should now show the letter ‘P’.

Evaluate
“Hang on”, you might say. “The alphabet doesn’t end with ‘P’. There should be more letters”. Rightly so. You remember that the B4 is a 4 bit computer and thus limited to the numbers 0 to 15? Our alphabet, however, has 26 different characters. How can we distinguish 26 different characters with 4 bit? We could add a 5th bit to the B4, but this would mean changing every single module. Another way is to use two 4 bit numbers. And this is where the left Variable comes into play. So we have created pages of 16 characters each. The left Variable will choose the page, and the right Variable will select the character, number or symbol from within that page. So, the first character on page 0 is an ‘A’. We have already seen this.

Mission 1.5
Set the right Variable to B0000 and the left Variable to B0001 (page 1)

Observe 1.5
What is the output of the Dot Matrix Display?

Compare 1.5
The Display should now show the letter ‘Q’.

Mission 1.6
Set the right Variable to B0001 and leave the left Variable to B0001 (page 1)

Observe 1.6
What is the output of the Dot Matrix Display?

Compare 1.6
The Display should now show the letter ‘R’.

Evaluate
he Dot Matrix Display has 10 pages of letters, numbers and symbols. You will find each of them in Appendix A. The following table provides a brief overview of what you get from each page:

Page Number	Content
B0000	Letters A ... P
B0001	Letters Q ... Z
B0010	Numbers 0 ...15
B0011	Symbols, including smileys
B0100	Flying Arrow
B0101	Dancing Dots
B0110	Pixel Mode
B0111	Ones digit of input number
B1000	Tens digit of input number
B1111	Display diagnostics

Summary of the Dot Matrix Display Output

Characters A ... P on *Page 0 (see Appendix A)*

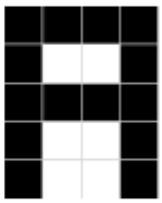
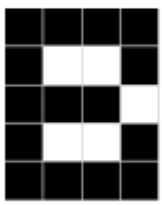
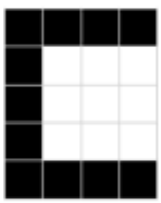
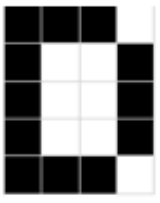
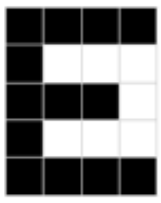
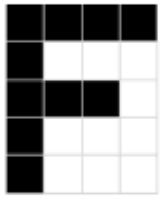
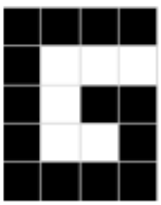
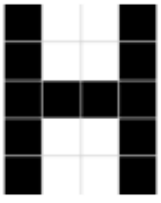
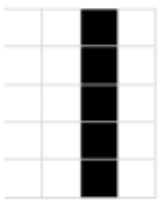
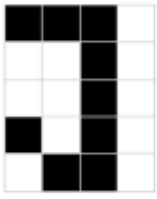
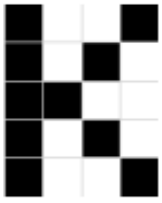
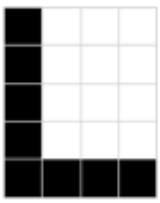
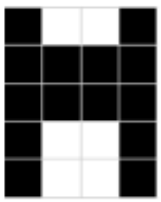
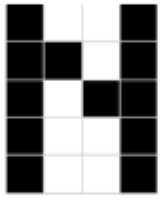
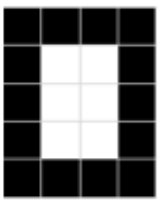
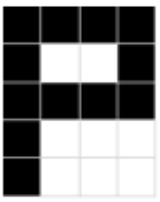
Mission 1.7

Go and mission with the Variables and try to generate each pattern from Appendix A on the Dot Matrix Display. Have fun !

Evaluate Deeper

How does it work?
When we send the page and data values to the Dot Matrix Display, a little microprocessor will decide which pattern to draw. It will lookup the pattern in its program and send electric impulses to the LEDs. Those that receive electricity will then light up.

So we are not actually sending a character to the Dot Matrix Display, but a reference to the character, which is already stored in the memory of the Dot Matrix Display. This is called a **Data Pointer**. The processor on the Dot Matrix Display is called a **Graphics Processor**, because it deals exclusively with graphics content.

	Page: B000			
Data	B0000	B0001	B0010	B0011
Pattern				
Data	B0100	B0101	B0110	B0111
Pattern				
Data	B1000	B1001	B1010	B1011
Pattern				
Data	B1100	B1101	B1110	B1111
Pattern				

Mission 2: Automating Graphics Output

Modules Required: Program Counter, Data RAM, Variable (Variable), Dot Matrix Display.

Analyse

In mission 1 we familiarised ourselves with the Dot Matrix Display module. In this mission, we want to output a word on the Dot Matrix Display. Our aim is to output the word ‘HELLO’ automatically. Let’s envisage an advertising banner of a shop. There is no one standing behind the banner turning a knob of something like a Variable.

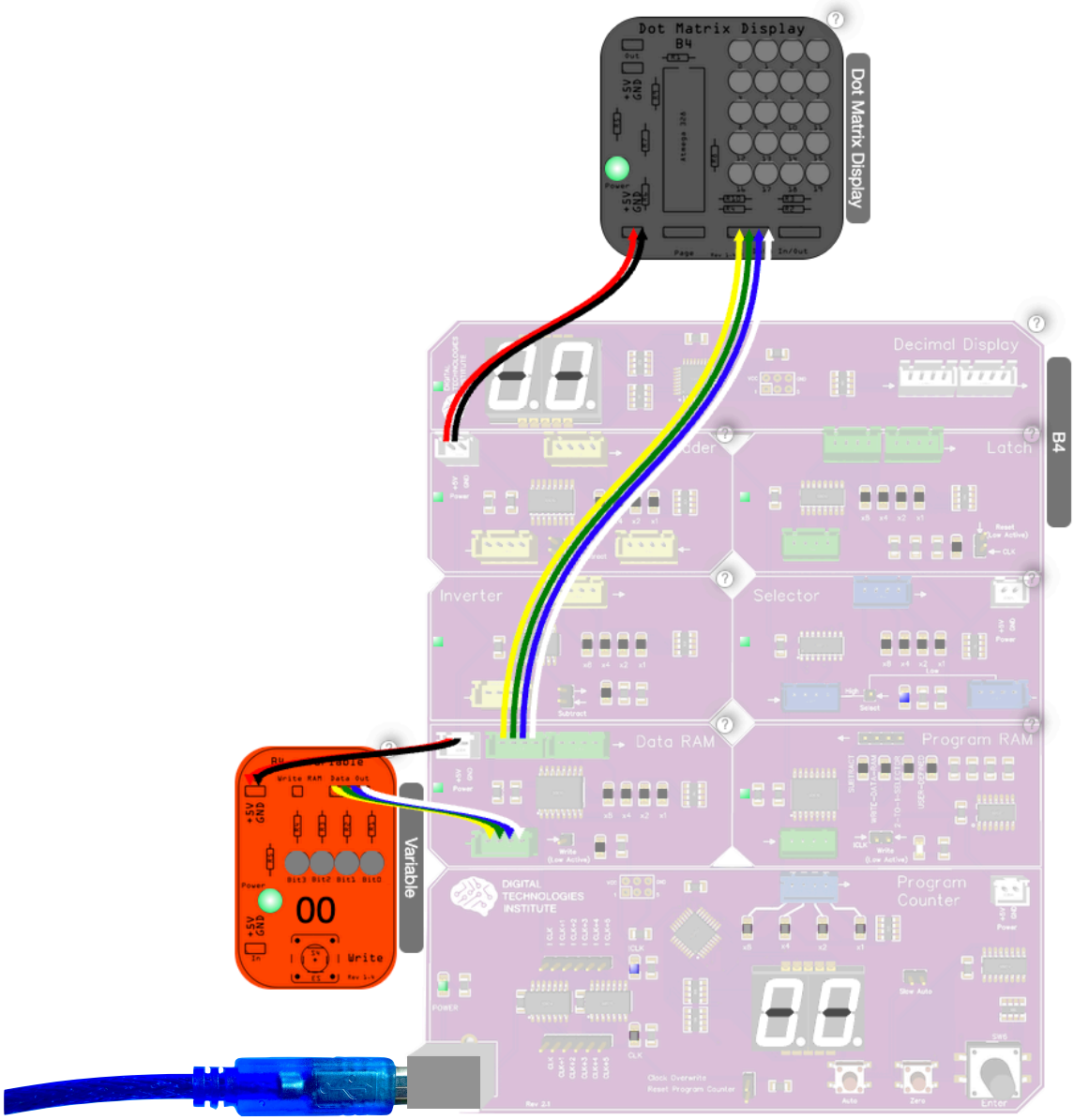
Design

Our objective is to program the Dot Matrix Display to output the letters ‘HELLO’ one after another. Do do this we enlist the services of our trusted Data RAM module, which we program with the data values of the letters we want the Dot Matrix Display to show.

Build

Let’s get started by assembling our mission according to the next figure:

1) Setup and Power
   A. Place the four modules on the desk in front of you as shown in the following diagram.
   B. Connect the Data RAM and Variable to the Program Counter with power wires as shown.
   C. Use a power wire and connect the left Data RAM with the Dot Matrix Display.
2) Data
   A. Connect the Output of the Program Counter to the PC In port of the Data RAM.
   B. Connect the 4 pin output of the Variable to the Data In connector of the Data RAM.
   C. Connect the 4 pin output of the Data RAM to the Data In/Out port of the Dot Matrix Display. There are two Data In/Out ports on the Dot Matrix Display. You can use either.
3) Control
   A) Connect a 1 pin wire from the Variable to the Write Data RAM In pin of the Data RAM. This wire is important to program the Data RAM.



Setup of Mission 2

Design

If nothing is connected to the Page input of the Dot Matrix Display, it will automatically select page B0000. That's fine for this mission since all the letters in 'HELLO' are on page B0000. In the next mission we will extend our setup to access multiple pages of characters. Ok, back to this mission.

You have completed the setup of the modules and checked the wiring? Very good. You probably remember how we programmed the Data RAM. If not, it is a good idea to refresh you knowledge by visiting the B4 Base Kit handbook and read mission 5. We will now design a program and then store it in the Data RAM module with the help of the Variable. For this, we re-use our programming table that you are already familiar with. Because there is no Program RAM in this mission, we simply delete the corresponding columns. We then get something like the following table:

	Data RAM				Description
Step #	**3**	**2**	**1**	**0**	
Step 15					
Step 14					
Step 13					
Step 12					
Step 11					
Step 10					
Step 9					
Step 8					
Step 7					
Step 6					
Step 5					
Step 4					
Step 3					
Step 2					
Step 1					
Step 0					

Programming Template for Mission 2

Design (code)
We begin by entering the character codes for the word ‘HELLO, beginning with the letter ‘H’ which we put at step 0. ‘H’ is B0111 (7). ‘E’ is B0100 (4), ‘L’ is B1011 (11) and ‘O’ is B1110 (14). We obtain these value from Appendix A. We then enter them into our table, remembering that the letter L occurs twice. This will result in the following table:

	Data RAM				Description
Step #	**3**	**2**	**1**	**0**	
Step 15					
Step 14					
Step 13					
Step 12					
Step 11					
Step 10					
Step 9					
Step 8					
Step 7					
Step 6					
Step 5					
Step 4	1	1	1	0	O
Step 3	1	0	1	1	L
Step 2	1	0	1	1	L
Step 1	0	1	0	0	E
Step 0	0	1	1	1	H

Program for Mission 2

Build (code)

Let's now program this data into the Data RAM module:

1) Press the Zero button on the Program Counter module to set it to step 0000. We call this resetting the Program Counter.
2) Set the data on the Variable to B0111 (letter 'H').
3) Press the button on the Variable to store the data. The Data RAM's LEDs will now display B0111 and you will see the letter 'H' on the Dot Matrix Display.
4) Press the Enter button on the Program Counter to advance it to step 0001.
5) Set the data on the Variable to B0100 (letter 'E').
6) Press the button on the Variable to store the data. The Data RAM's LEDs will display B0100 and you will see the letter 'E' on the Dot Matrix Display.
7) Continue with these steps until you have programmed the letter 'O' in step 4 of the program.

Mission

When you have completed the programming, press the Reset button on the Program Counter. The Dot Matrix Display will show ‘H’ and the Program Counter 00. Now press the Enter button on the Program Counter. You should now see a ‘E’ on the Dot Matrix Display. Press the button again and you see a ‘L’, then ‘L’ again and finally ‘O’ when you have reached step 04 on the Program Counter.

Well done. You have programmed your first graphical output.

Observe

What will happen when we press the Enter button again?

Compare

We haven’t programmed anything into the memory of our Data RAM for position 05 (B0101). If it contains B0000, the Display will show the letter ‘A’. Any other binary value will still show some letter. And if you inspect the character table for page B0000 in Appendix A carefully you will notice that it does not contain an empty character (space). So the display will output all sorts of useless character for positions 5 to 15, which is not what we want.

Evaluate

Computers have no sense of ‘no data’. Anything that consists of 1’s and 0’s is data for a computer, even B0000. If we don’t want an output, we have to point the computer to an area of memory that has been designated as producing empty space.

If you have a look at Appendix A again, you will see that page B0001 contains an empty space character at position B1010. If we could quickly switch to page B0001 we could access the space character. That’s what we will do in the next mission.

Mission 3: Do-Re-Mi[1]

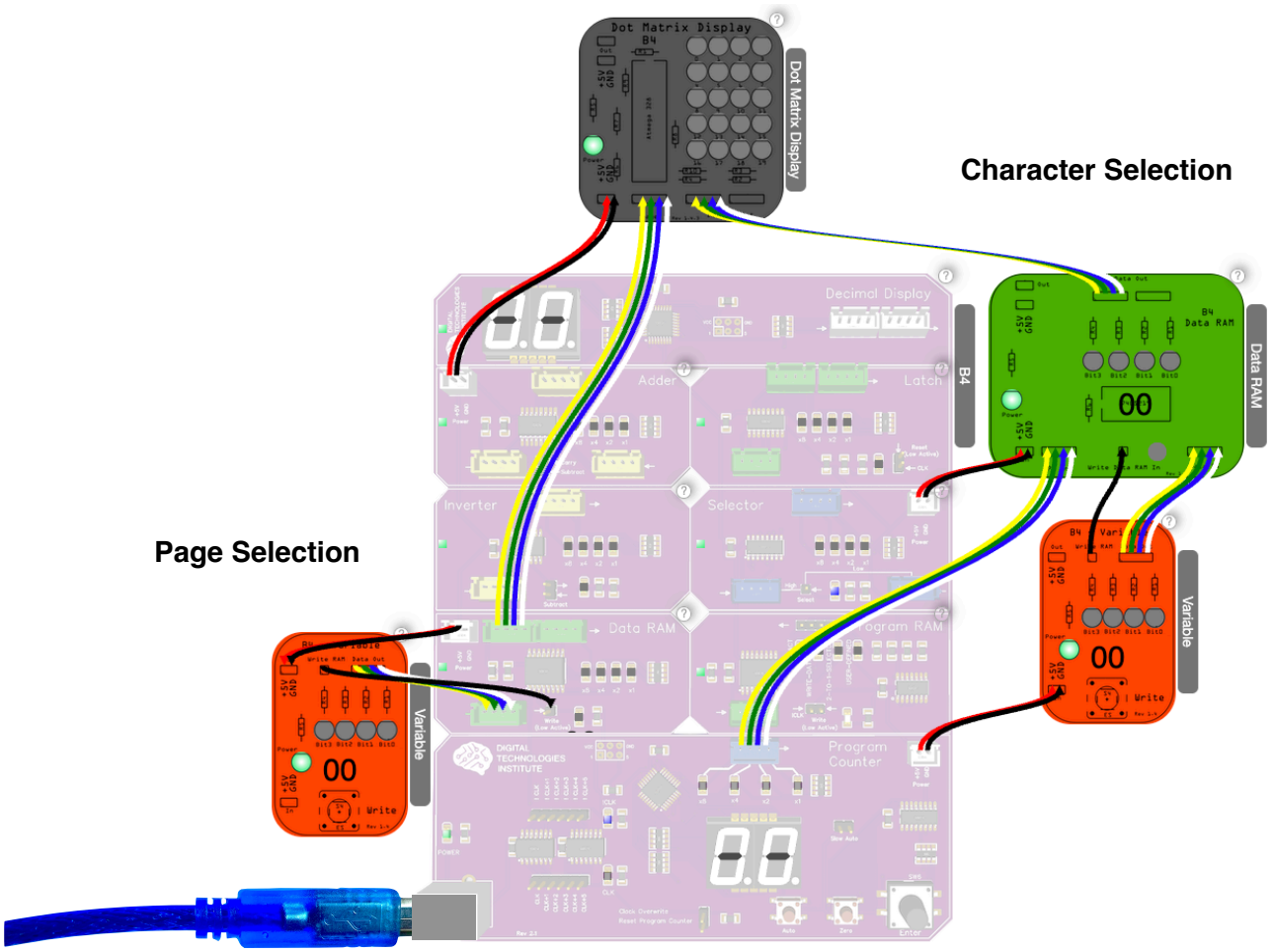
Modules Required: Program Counter, 2x Variable, 2x Data RAM, Dot Matrix Display

Analyse
In the famous musical, The Sound of Music, Maria teaches the notes of the major musical scale to the Von Trapp children. She sings: “When you know the notes to sing You can sing most an--ny--thing”. Similarly, once we can access all the letters of the alphabet we can write any word (that’s not longer than 16 characters in the case of a 4 bit computer). In the previous mission we were limited to accessing the characters ‘A’ to ‘P’ from within a single page. What could we do to overcome this limitation? Any ideas?

Design
We could: - Add a Variable to the page input of the Dot Matrix Display like in mission 1. - Add a second Data RAM module in which we program the page number we want to access. Let’s combine these ideas: The second Data RAM module will hold the pages information which we will program into it with a Variable. We already know how to add a Data RAM module to our setup from the previous mission. All we need is a second Data RAM module and another Variable.

Build
We re-use our setup from the previous mission and add a Data RAM module and a Variable to the left of our setup as shown in the Figure below. We then connect power, data and control wires as shown.

[1] Famous song from ‘The Sound of Music’



Setup of Mission 3

Build (continued)
The left Variable and Data RAM will be in charge of the page selection, whilst the right Variable and Data RAM will be in charge of the character selection. With character we mean any kind of letter, number or symbol. We then extend our graphics program template by adding columns for the page information as follows. The information for the page will reside in the Page RAM on the left and the character information will be on the right, in the Character RAM. This mirrors our physical setup in the mission.

	Page RAM				Character RAM				Description
Step #	**3**	**2**	**1**	**0**	**3**	**2**	**1**	**0**	
Step 15									
Step 14									
Step 13									
Step 12									
Step 11									
Step 10									
Step 9									
Step 8									
Step 7									
Step 6									
Step 5									
Step 4									
Step 3									
Step 2									
Step 1									
Step 0									

Design (code)

Very good. Now let's enter our program from mission 2 into the new template and also fill in the information from which page we want to load each character. In our example of the word 'HELLO' all characters can be found on page B0000.

	Page RAM				Character RAM				Description
Step #	**3**	**2**	**1**	**0**	**3**	**2**	**1**	**0**	
Step 15									
Step 14									
Step 13									
Step 12									
Step 11									
Step 10									
Step 9									
Step 8									
Step 7									
Step 6									
Step 5									
Step 4	0	0	0	0	1	1	1	0	O
Step 3	0	0	0	0	1	0	1	1	L
Step 2	0	0	0	0	1	0	1	1	L
Step 1	0	0	0	0	0	1	0	0	E
Step 0	0	0	0	0	0	1	1	1	H

Design (code) (continued)

So far so good. We now want to clear the Dot Matrix Display after our text has been shown. We do this by selecting the space character B1010 from page B0001 (again, have a look at Appendix A). This will result in the following table:

	Page RAM				Character RAM				Description
Step #	**3**	**2**	**1**	**0**	**3**	**2**	**1**	**0**	
Step 15	0	0	0	1	1	0	1	0	Space
Step 14	0	0	0	1	1	0	1	0	Space
Step 13	0	0	0	1	1	0	1	0	Space
Step 12	0	0	0	1	1	0	1	0	Space
Step 11	0	0	0	1	1	0	1	0	Space
Step 10	0	0	0	1	1	0	1	0	Space
Step 9	0	0	0	1	1	0	1	0	Space
Step 8	0	0	0	1	1	0	1	0	Space
Step 7	0	0	0	1	1	0	1	0	Space
Step 6	0	0	0	1	1	0	1	0	Space
Step 5	0	0	0	1	1	0	1	0	Space
Step 4	0	0	0	0	1	1	1	0	O
Step 3	0	0	0	0	1	0	1	1	L
Step 2	0	0	0	0	1	0	1	1	L
Step 1	0	0	0	0	0	1	0	0	E
Step 0	0	0	0	0	0	1	1	1	H

Build (code)

Use the left Variable to program the RAM module holding the page data (Page RAM) and the right Variable to program the RAM module holding the character data (Character RAM). The programming process is very similar to the one described in the previous mission:

1) Press the button on the Program Counter module until it is at step 0000.
2) Set the data on the right Variable to B0111 (letter ‘H’).
3) Set the data on the left Variable to B0000 (page 0).
4) Press the button on the right Variable to store the page data.
5) Press the button on the left Variable to store the character data.
6) Press the Enter button on the Program Counter to advance to step 01.
7) Repeat the above steps with the information from the above table until you have again reached step 0 on the Program Counter.

Mission
1) When you have completed the programming, the Dot Matrix Display will show ‘H’ and the Program Counter 00. 2) Now press the Enter button on the Program Counter. You should now see the letter ‘E’ on the Dot Matrix Display. 3) Press the button again and you see a ‘L’, then ‘L’ again and finally ‘O’ when you have reached step 04 on the Program Counter. 4) If you press the Enter button again, the Display will go blank (empty) and stay so until you come back to step 0 (‘H’). Very good.

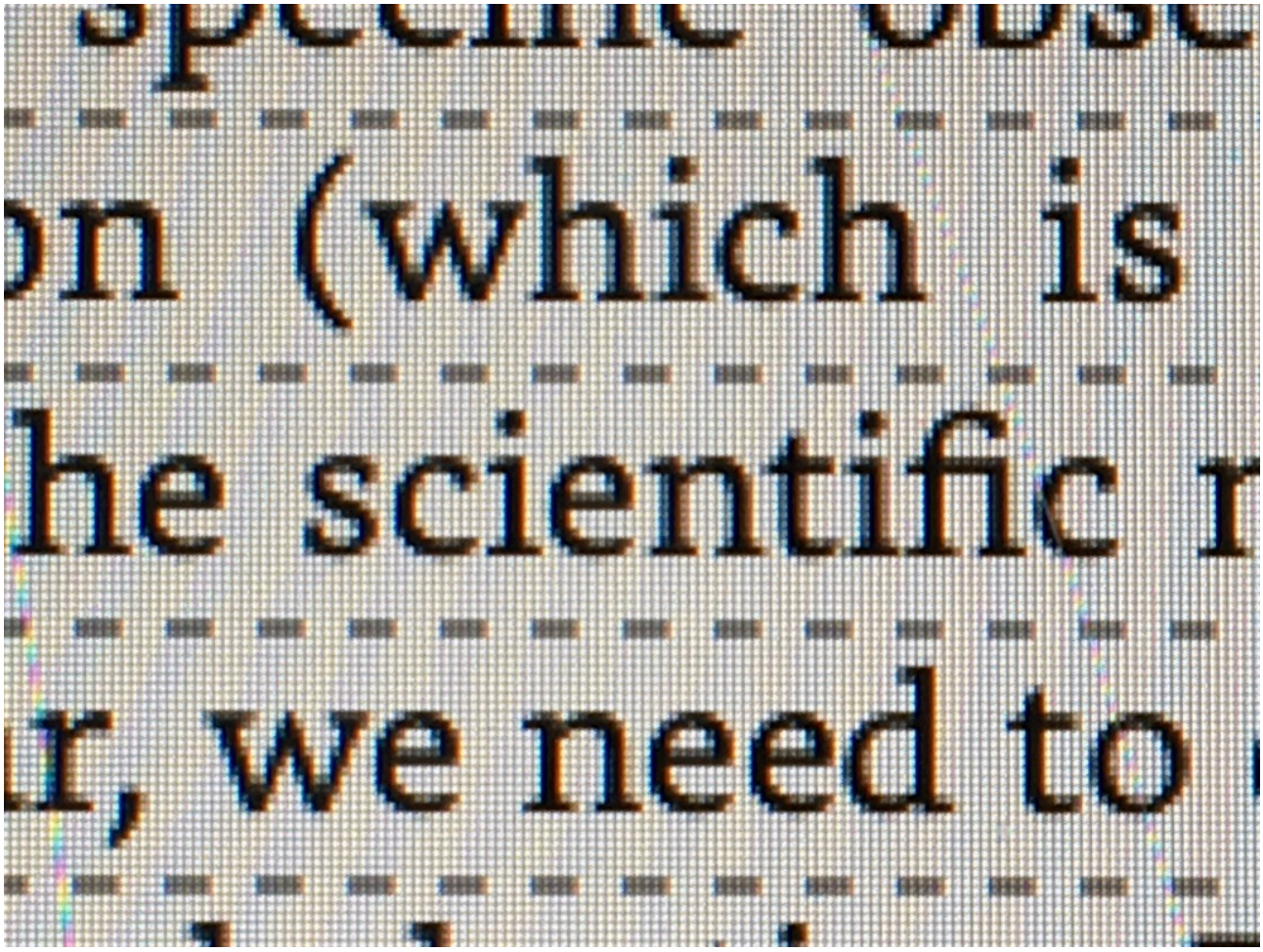
Now run a few more experiments:

Question 3.1	
?	Using the method from this mission, output a word or sentence of your choice on the Dot Matrix Display. Use characters from at least 3 different pages.
	Study the different characters, numbers and symbols in the tables listed in Appendix 1. Can you: a) produce an animation of a smiley that goes from happy to sad? b) produce an animation of an arrow that flies through the display from left to right and back? c) produce an animation of two dots that circle around each other?

Mission 4: Pixels

Modules Required: Same as in mission 3.

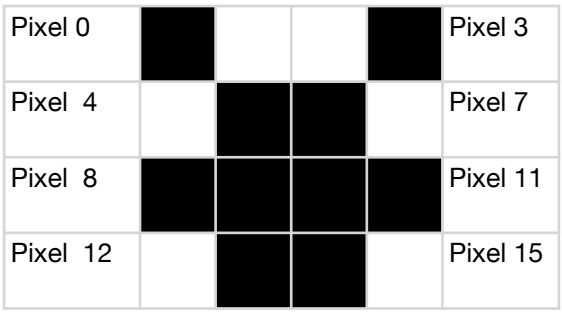
Analyse
In the previous mission, we used the various characters that have been pre-programmed into the Dot Matrix Module. However, the last page B0110 allows the direct access of individual LEDs of the Dot Matrix Display. Each of the LEDs in a display is called a pixel. According to our friends at Wikipedia, “The word pixel is based on a contraction of pix (from word "pictures", where it is shortened to "pics", and "cs" in "pics" sounds like "x") and el (for "element")[...]”[2]. A pixel is the smallest addressable point in a display device. All displays consist of pixels. The are so tiny that we can’t tell them apart with our naked eyes. However, when looked at under a magnifying glass, they do become visible. In the image below, we have taken a photo of an iPad 2 screen with an iPhone 6 camera. We can clearly see the grid of pixels on the iPad2’s screen.



Pixels of an iPad 2 Display

Analyse (continued)

The more pixels a display has, the higher is the resolution of that display. An tablet display has, for example, a resolution of 1024 pixels by 768 pixels, arranged in a grid as shown above. A tablet display therefore has 1024x768= 786,432 pixels that can be individually controlled. The more pixels a display has at a given display size, the higher the resolution of that display. High-res displays look sharper and can display more content than low-res displays. Our Dot Matrix Display is a low-res display, as it only has 5x4=20 pixels. But that's enough to run exciting experiments. The pixels 0-15 can be individually controlled when page B0110 is selected. Why only 16 pixels, and not 20? The data bus is only 4 bit wide and 4bit is just enough to distinguish 16 values.



Pattern for Mission 4

Design (code)

We want to draw the above teddybear face on our display.:

The pixels are numbered from 0 at the top-left corner to 15 at the bottom-right. Because only one pixel can be 'on' at any given time we will apply a trick to make our brain believe that multiple pixels are 'on' at the same time, thus giving the impression of an image. If you have ever watched a movie you probably know that a movie consists of pictures that are shown rapidly. Our eyes and brain are just not fast enough to see them individually and thus create the illusion of a 'motion picture'. We will do the same here. Let's first fill-in our programming template. Pixel 0 has the code B0000, pixel 1 is B0001 and so forth. Refer to Appendix A for details. Eventually we get the following table:

	Page RAM				Character RAM				Description
Step #	**3**	**2**	**1**	**0**	**3**	**2**	**1**	**0**	
Step 15	0	0	0	1	1	0	1	0	Space
Step 14	0	0	0	1	1	0	1	0	Space
Step 13	0	0	0	1	1	0	1	0	Space
Step 12	0	0	0	1	1	0	1	0	Space
Step 11	0	0	0	1	1	0	1	0	Space
Step 10	0	0	0	1	1	0	1	0	Space

	Page RAM				Character RAM				Description
Step #	**3**	**2**	**1**	**0**	**3**	**2**	**1**	**0**	
Step 9	0	1	1	0	1	1	1	0	Pixel 14
Step 8	0	1	1	0	1	1	0	1	Pixel 13
Step 7	0	1	1	0	1	0	1	1	Pixel 11
Step 6	0	1	1	0	1	0	1	0	Pixel 10
Step 5	0	1	1	0	1	0	0	1	Pixel 9
Step 4	0	1	1	0	1	0	0	0	Pixel 8
Step 3	0	1	1	0	0	1	1	0	Pixel 6
Step 2	0	1	1	0	0	1	0	1	Pixel 5
Step 1	0	1	1	0	0	0	1	1	Pixel 3
Step 0	0	1	1	0	0	0	0	0	Pixel 0

Program for Mission 4

Build (code)

We program the Page RAM and Character RAM modules with the same method we used on mission 3. If you have forgotten how we did this, you might want to flip back a couple of pages to refresh your memory.

Mission 4.1

Once the RAM modules have been programmed, set the Program Counter to step 0.

Observe 4.1

What do you see on the Dot Matrix Display?

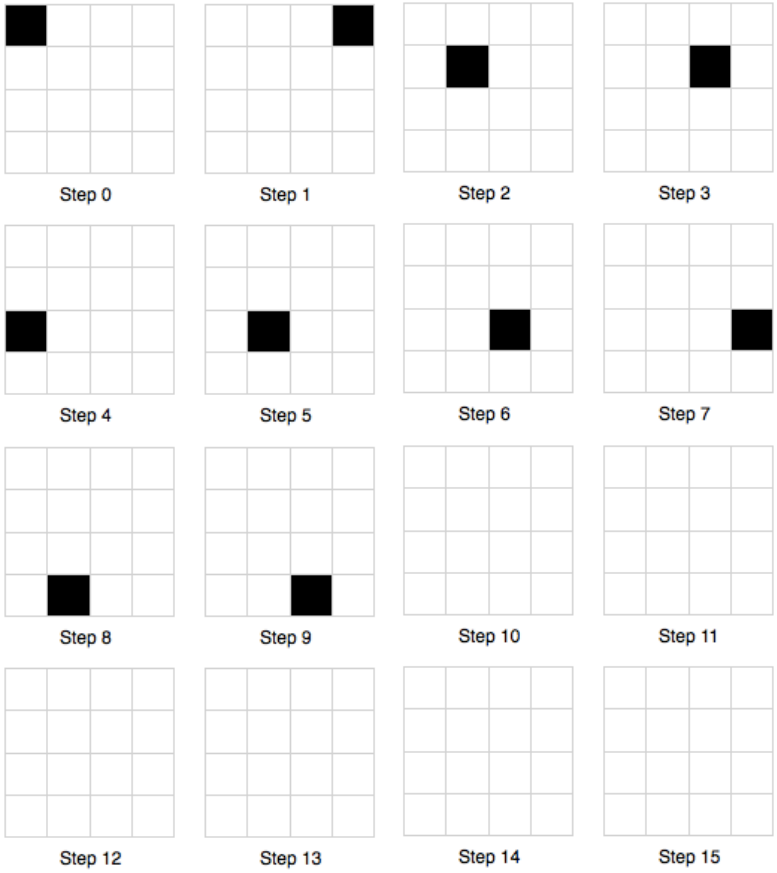
Compare 4.1

You should see the top left pixel ‘on’ and all other pixels are ‘off’.

Mission 4.2

Press the Enter button of the Program Counter: Pixel 0 turns off and pixel 3 lights up. Press the Enter button again: Pixel 5 lights up. The following figure shows which pixel will be on during each step of our program.

Evaluate



However, by pressing the Enter button repeatedly, we still only see individual pixels. Where is the teddybear face?

Mission 4.3

Have you noticed the button labelled **Auto** on the Program Counter? Press and hold it.

Observe 4.3

What happens?

Compare 4.3

The Auto(matic) button sends repeated fast Clock (CLK) signals to the Program Counter. Our program now executes much faster. As a result, we see the image. Though it is flickering a bit, it is clearly visible. If we increased the CLK speed further we would see a still image.
We talked about the Clock signal during mission 1 of the B4 base Kit Handbook. You might want to go there to refresh your memory.

Question 4.1	
?	If you chose a different order for the pixels in your program. Would you still see the same picture?
	Draw a picture of your own choice and program it into the RAM modules. Execute the resulting program

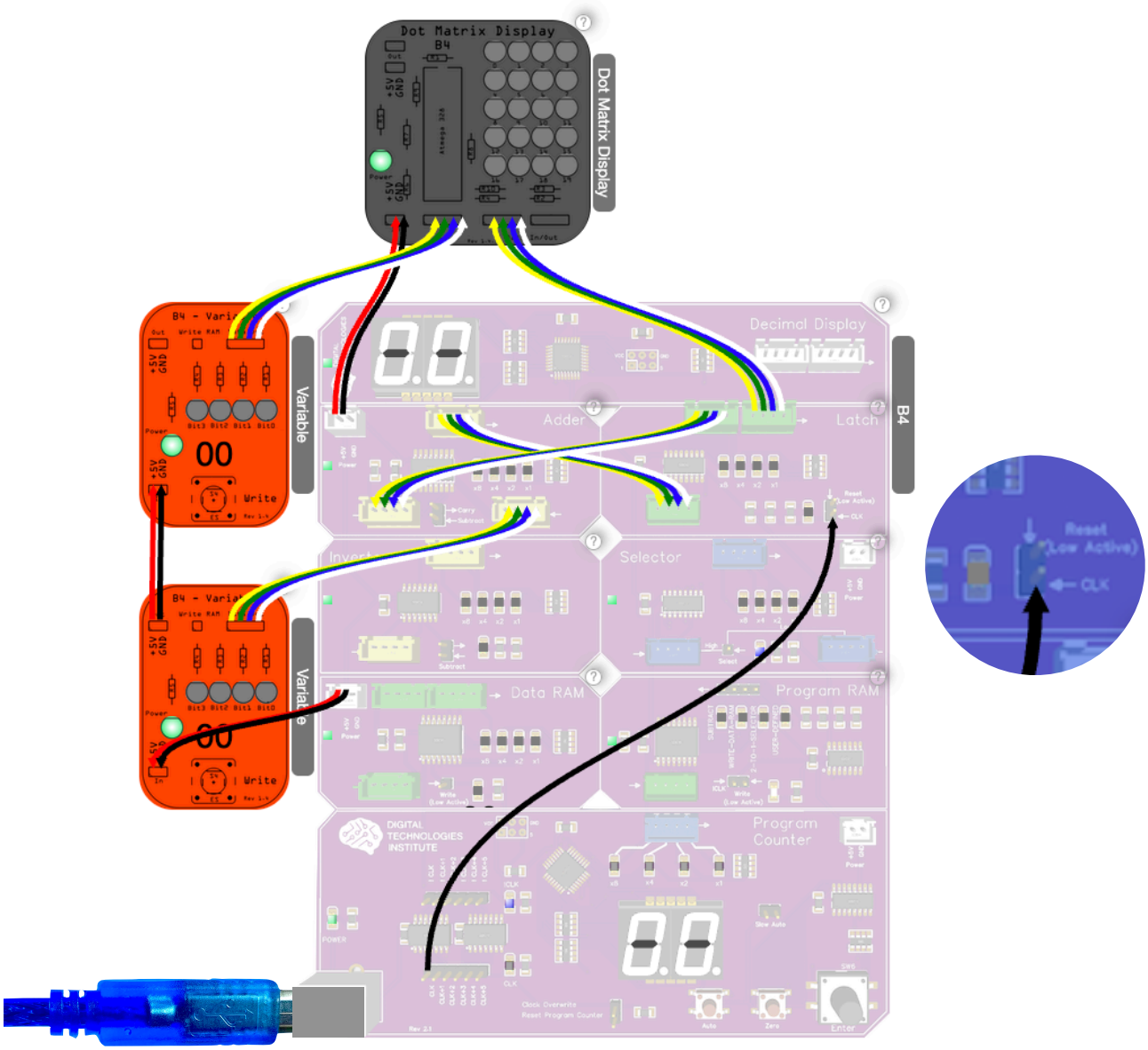
Evaluate
Summary In this mission we have learned that we can draw our own images by programming them pixel by pixel and executing the resulting program really fast with the Automatic button on the Program Counter. Although only one pixel is lit at any given time, our brain produces the illusion of an image consisting of multiple pixels.

Mission 5: Making Maths Artwork

Modules Required: Program Counter, Adder, 2x Variable, Latch, Dot Matrix Display.

Design
Our goal is to construct a simple machine that produces beautiful artwork. Let's combine our knowledge about pixels with what we already know about the adding machine from the B4 Spark Kit. Let's do an mission.

Build
We connect the modules as shown on the following page. Connect all the power and data wires and don't forget to run a 1 pin wire from one of the Program Counter's CLK pins to the CLK In pin of the Latch. The **top left Variable will be permanently set to B0110**, thus selecting the pixel mode on the Dot Matrix Display. The Adder will add two numbers: One from the bottom Variable and the number coming from the Latch output. The Latch will remember the result of any calculation. **Before you try this mission, make sure you reset the Latch by temporarily connecting another 1 pin wire from 'Latch Reset In' to GND. Then disconnect the wire before you continue**.

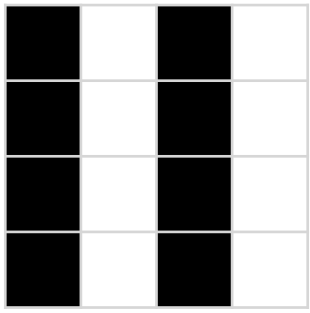


Setup of Mission 5

Mission 5.1
Set the bottom Variable is to 2 (B0010) and then press and hold the Automatic button on the Program Counter.

Observe 5.1
What do you see on the Dot Matrix Display?

Compare 5.1
In rapid Automatic mode, the pattern on the Display will look like this:



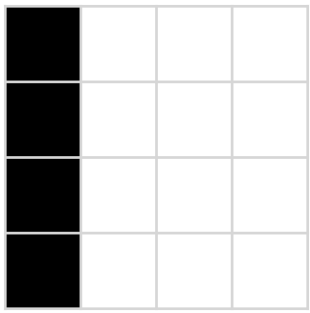
2s Pattern

Evaluate
The bottom Variable is set to 2 (B0010) The Latch has an initial value of B0000. On the first CLK cycle, the Adder will add 2+0=2, which will be remembered by the Latch. Pixel 2 lights up on the Dot Matrix Display. In the next CLK cycle, the Adder will add 2 (from the Variable) and 2 from the Latch: 2+2=4. Again, the Latch will sore the result and the Dot Matrix Display will activate Pixel number 4. With this, we will eventually produce the sequence 0, 2, 4, 6, 8, 10, 12, 14, 0, 2, 4, ... and light up the corresponding pixels on the display. This will look like two parallel lines.

Mission 5.2
After you have run the +2 mission, reset the mission by resetting the Program Counter and the Latch and then set the bottom Variable to 4. Then, press and hold the Automatic button on the Program Counter.

Observe 5.2
How will the pattern look this time?

Compare 5.2
You will get the following sequence: 0,4,8,12,0, ... which looks like this. One vertical line.

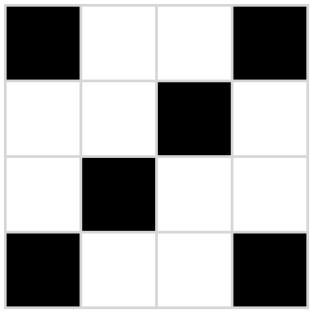


4s Pattern

Mission 5.3
Reset the mission by resetting the Program Counter and the Latch and then set the bottom Variable to an uneven number, such as 3. Then, press and hold the Automatic button on the Program Counter.

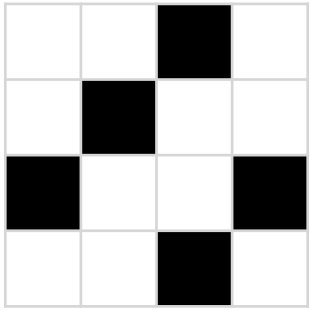
Observe 5.3
We expect this sequence: 0,3,6,9,12,15,2,5,8,11,14,1,4,7,10,13, 0,3,6,9,... So this pattern will only repeat every third cycle.

Compare 5.3
The first part of the pattern (0,3,6,9,12,15) looks like this:



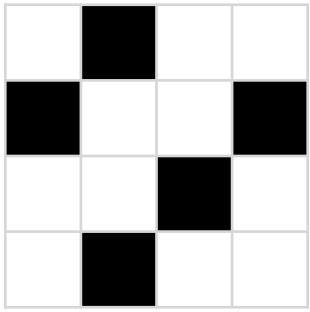
3's pattern 0 to 15.

The second part of the pattern (2,5,8,11,14) looks like this:



3's pattern 2 to 14.

And finally, the third part of the pattern (1,4,7,10,13) looks like this:



3's pattern 1 to 13.

Evaluate
This gives the illusion of a moving diagonal from the bottom-right corner to the top left corner of the Display. Isn't this wonderful? All that really happens is that pixels are switched on and off. Because this is happening very fast, our brain tries to make sense out of what it sees and decides that it must be seeing something that moves.

If things move too fast for you, use the Enter button instead of the Automatic button to see step by step how the pattern is changing.

Question 5.1	
?	Why does the pattern not change when the bottom Variable is set to zero (B0000)?
	Explain why the pattern seems to be moving down when the bottom Variable is set to B0001, but moving up when the value is B1111.
	Whilst the program is running, you can try to turn the bottom Variable and observe the results on the Dot Matrix Display. Try to produce different patters, observe similarities and try to explain what you see with what is happening inside the machine.

Further Reading

Below, we have listed some really good resources that we used during the design of the B4. We very much recommend reading them.

Charles Petzold, CODE The Hidden Language of Computer Hardware and Software, 1999
http://www.charlespetzold.com/code/
ASCII Table and Description http://www.asciitable.com
ASCII https://en.wikipedia.org/wiki/ASCII
Pixel: https://en.wikipedia.org/wiki/Pixel
Pixel Art https://en.wikipedia.org/wiki/Pixel_art

Troubleshooting

Every good mission has the potential for failure. This is usually the moment when we learn something new. Below is a list of the typical errors and their solutions.

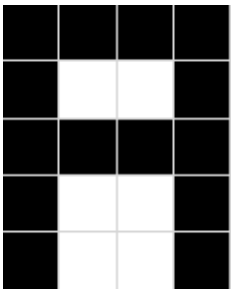
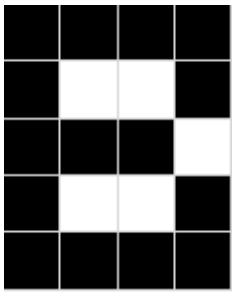
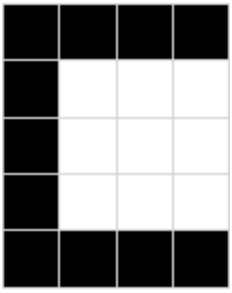
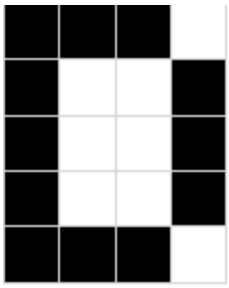
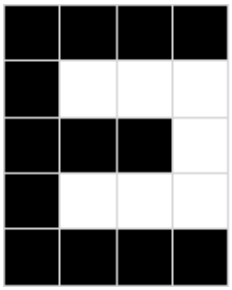
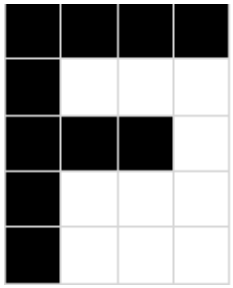
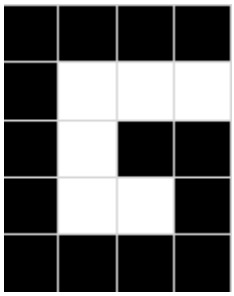
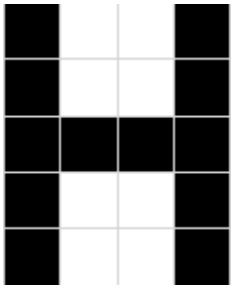
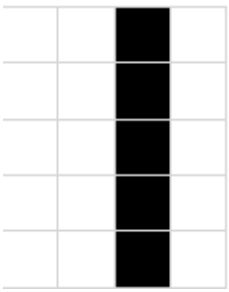
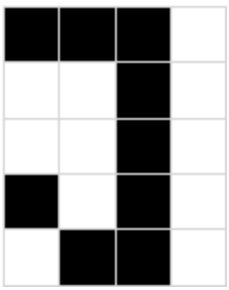
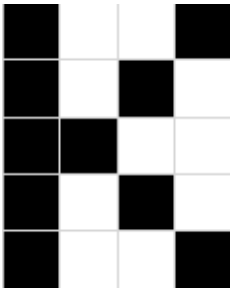
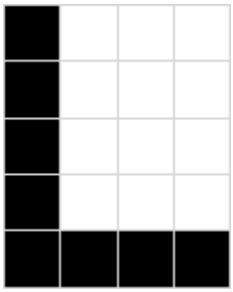
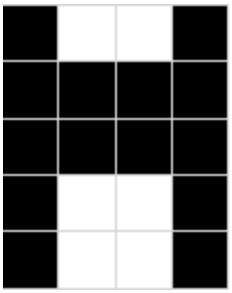
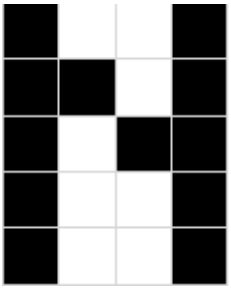
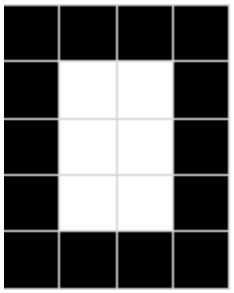
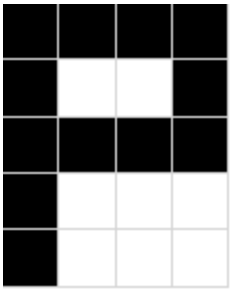
Symptom	Solution
Green light of a module is off	Check if power cable is connected
	Check if the wires at the power cable plugs are fully inserted. Change cable.
Unexpected behaviour. Odd output of the modules. Looks erratic.	Check if all wires are properly connected. Tick them off one by one on the schematic of the corresponding mission.
	Check if the wires at the data cable sockets are fully inserted. Change cable.
	Have you inserted the correct module? Check!
All lights are off	Connect USB cable to a computer, USB power outlet or USB battery.
	There may be a short circuit, usually caused by a power cable. Disconnect all power cables from the Program Counter and check if the Program Counter's green LED comes on. If yes, carefully connect one module after the other.
Dot Matrix Module works, but shows the wrong pattern	Check that all data wires are connected to the correct port of the display.
Dot Matrix Module behaves erratically.	Run the Dot Matrix Display's self-diagnostic program by selecting page B1111.

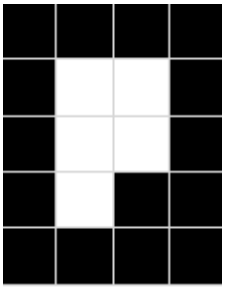
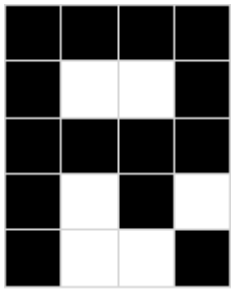
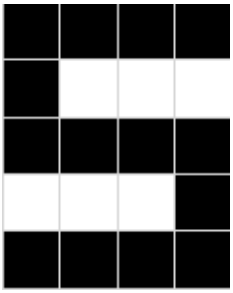
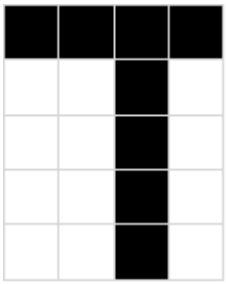
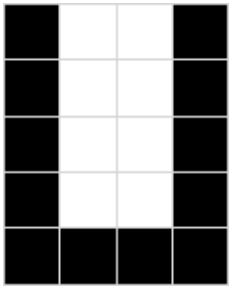
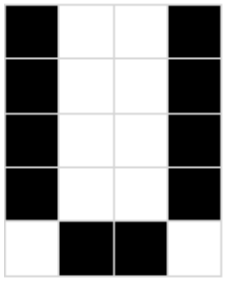
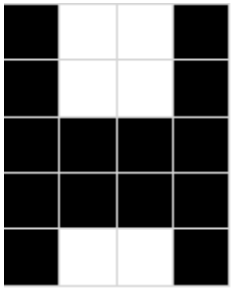
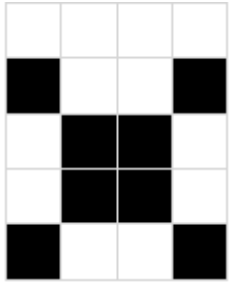
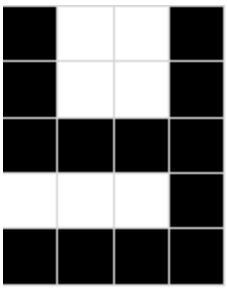
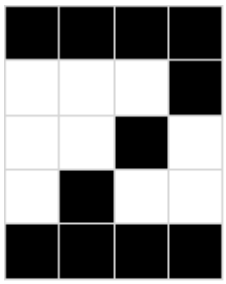
Still got problems? Email us at: enquiries@digital-technologies.institute.

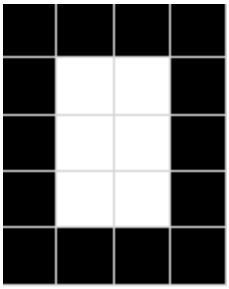
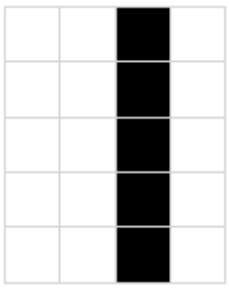
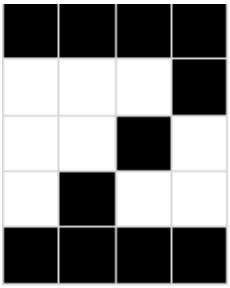
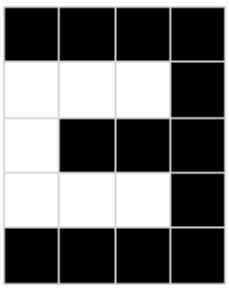
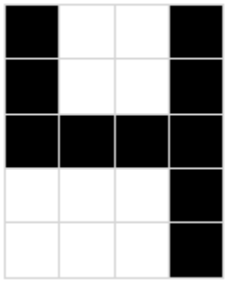
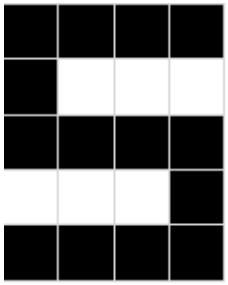
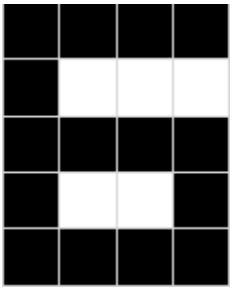
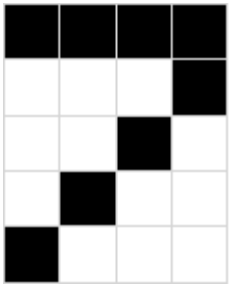
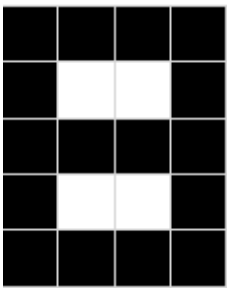
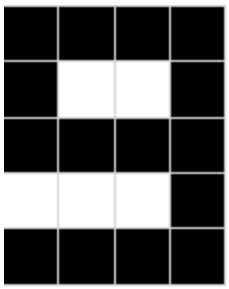
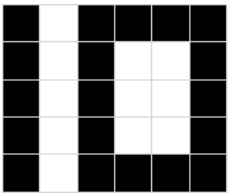
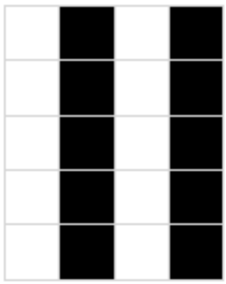
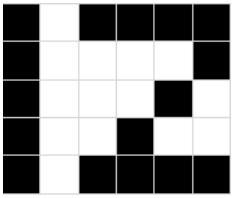
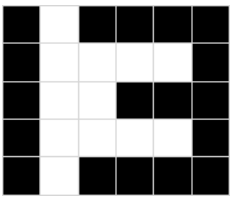
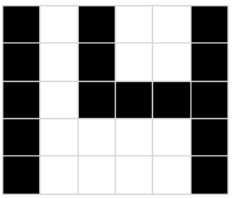
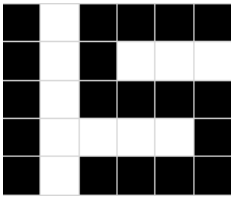
Appendix A: Character Tables

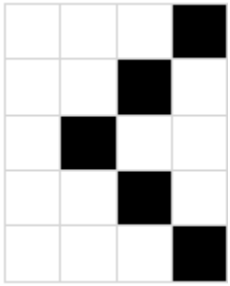
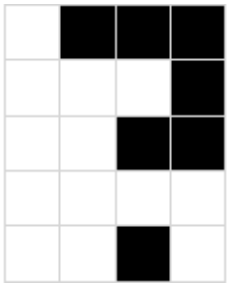
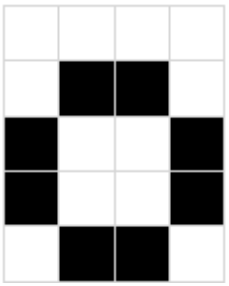
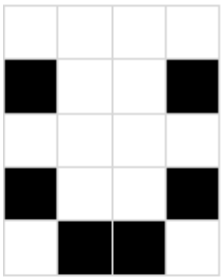
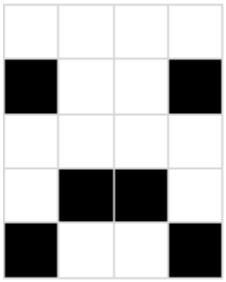
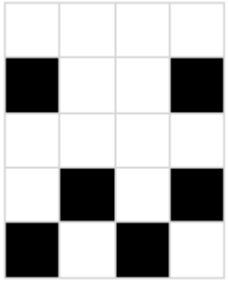
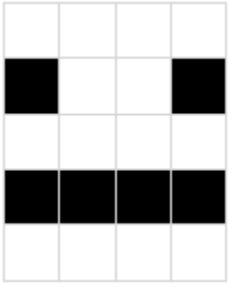
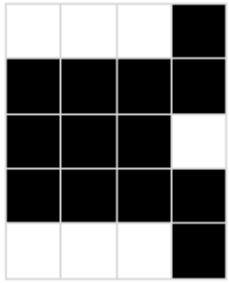
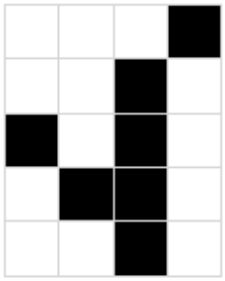
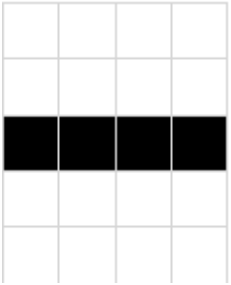
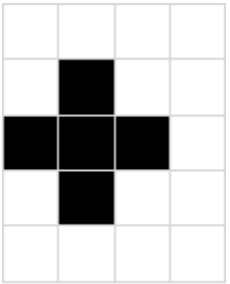
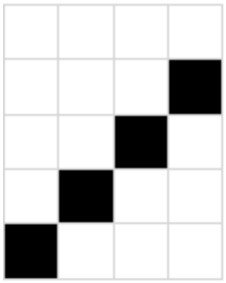
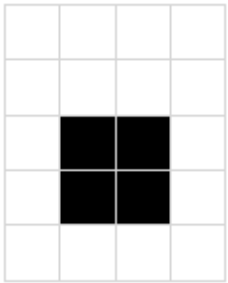
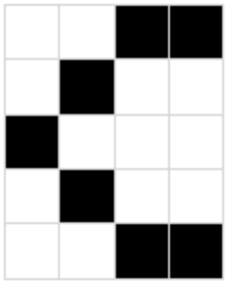
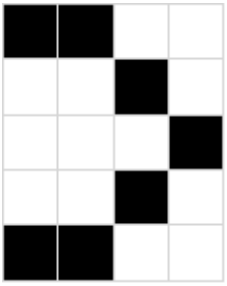
The Dot Matrix Display has 10 pages (B0000 to B0110) of letters, numbers and symbols. The following table provides a brief overview of what you get from each page. You will find each of them on the following pages.

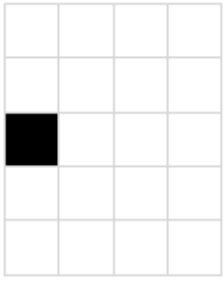
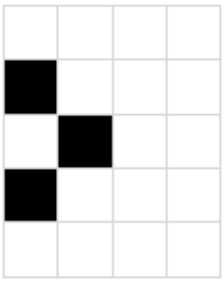
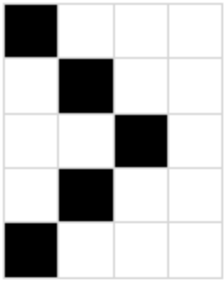
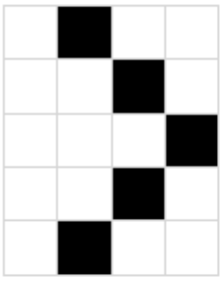
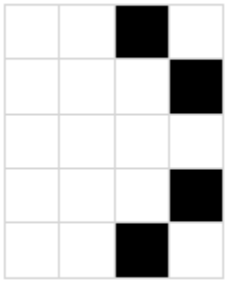
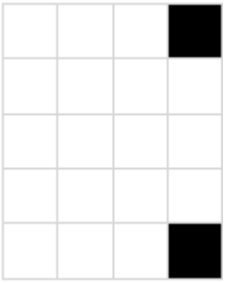
Page Number	Content
B0000	Letters A ... P
B0001	Letters Q ... Z
B0010	Numbers 0 ...15
B0011	Symbols, including smileys
B0100	Flying Arrow
B0101	Dancing Dots
B0110	Pixel Mode
B0111	Ones digit of input number
B1000	Tens digit of input number
B1111	Display diagnostics

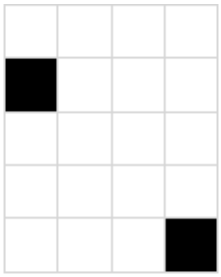
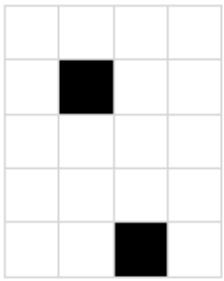
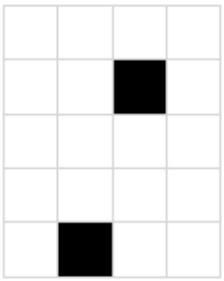
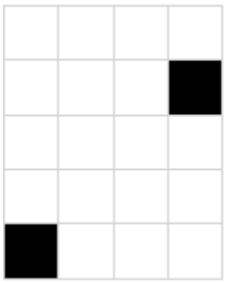
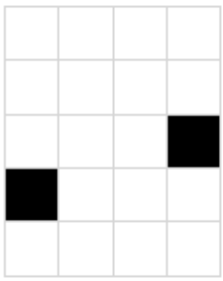
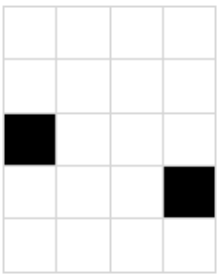
	Page: B0000			
Data	B0000	B0001	B0010	B0011
Pattern				
Data	B0100	B0101	B0110	B0111
Pattern				
Data	B1000	B1001	B1010	B1011
Pattern				
Data	B1100	B1101	B1110	B1111
Pattern				

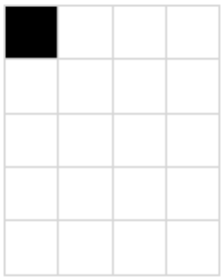
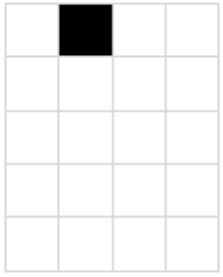
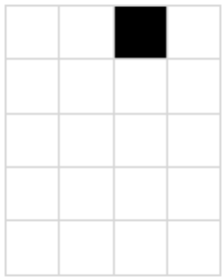
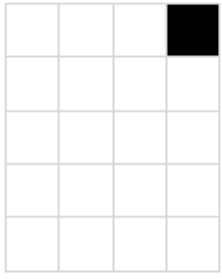
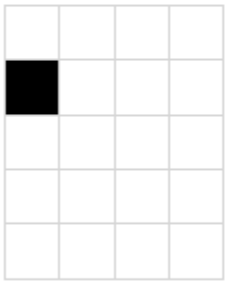
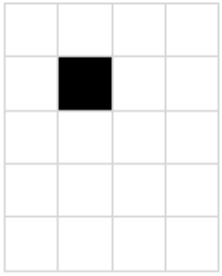
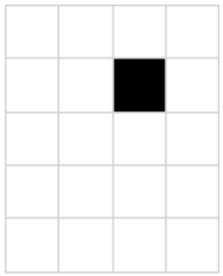
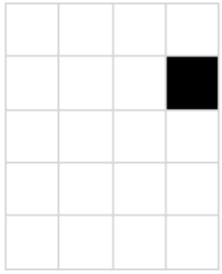
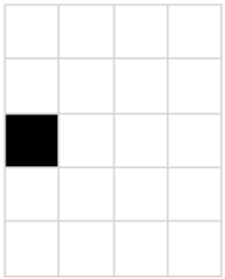
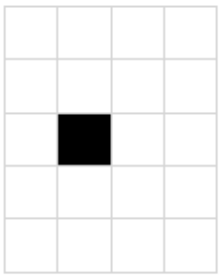
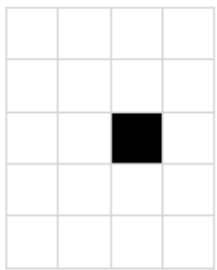
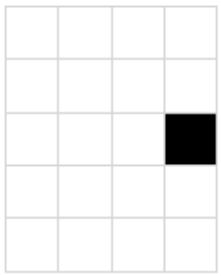
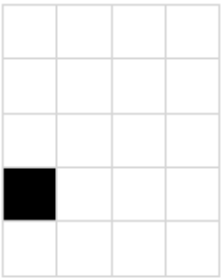
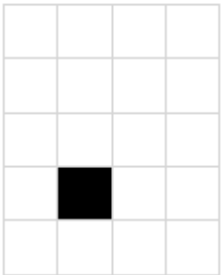
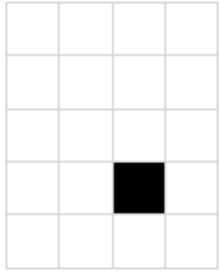
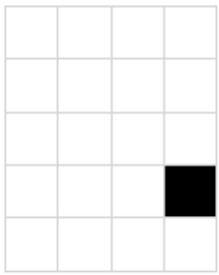
	Page: B0001			
Data	B0000	B0001	B0010	B0011
Pattern				
Data	B0100	B0101	B0110	B0111
Pattern				
Data	B1000	B1001	B1010	B1011
Pattern				
Data	B1100	B1101	B1110	B1111
Pattern				

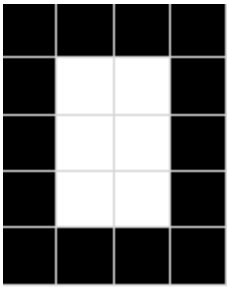
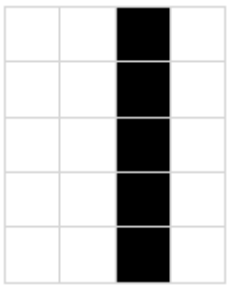
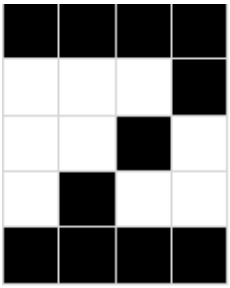
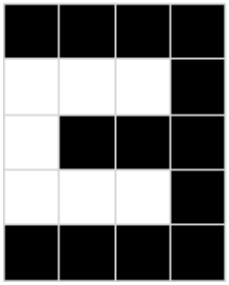
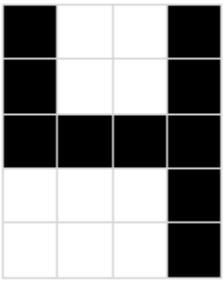
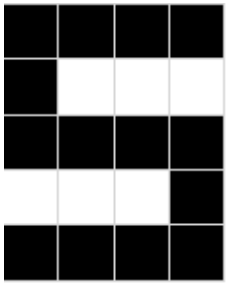
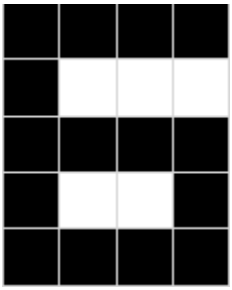
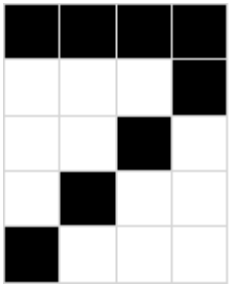
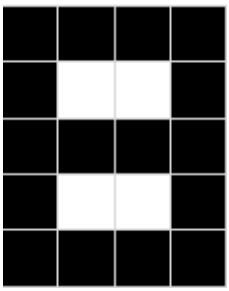
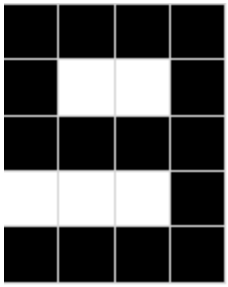
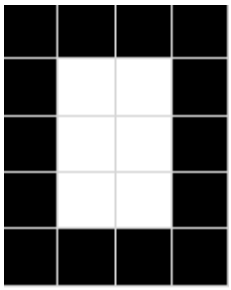
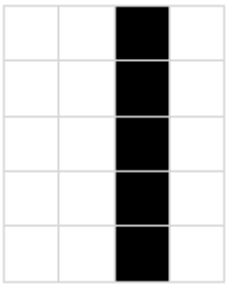
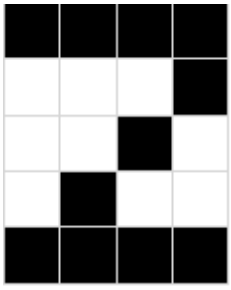
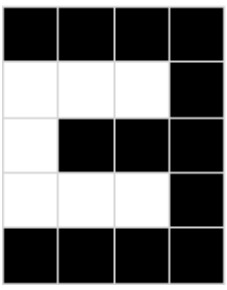
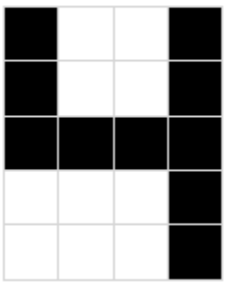
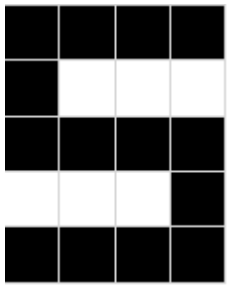
	Page: B0010			
Data	B0000	B0001	B0010	B0011
Pattern				
Data	B0100	B0101	B0110	B0111
Pattern				
Data	B1000	B1001	B1010	B1011
Pattern			 scrolling	
Data	B1100	B1101	B1110	B1111
Pattern	 scrolling	 scrolling	 scrolling	 scrolling

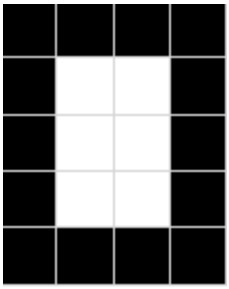
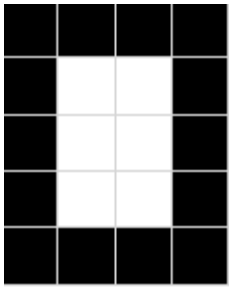
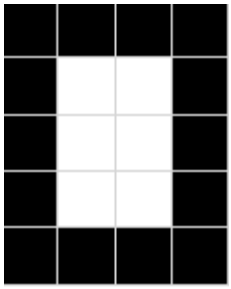
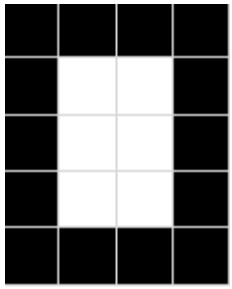
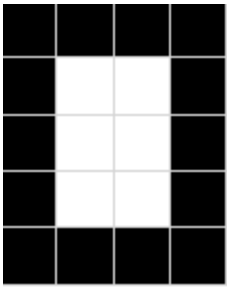
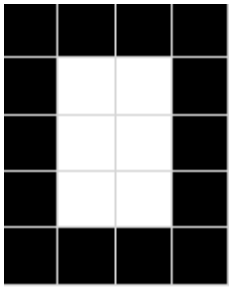
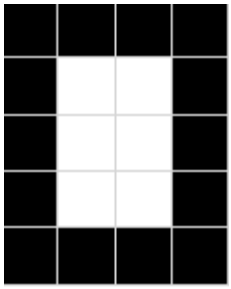
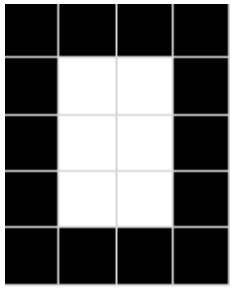
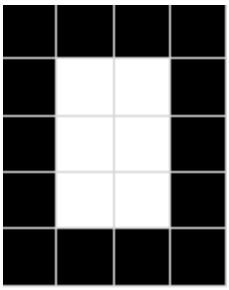
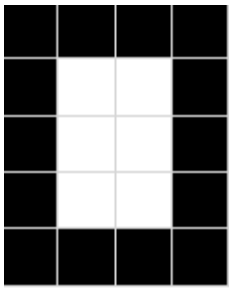
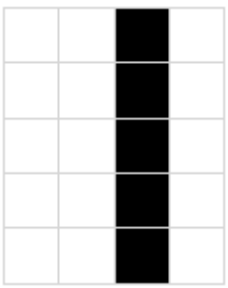
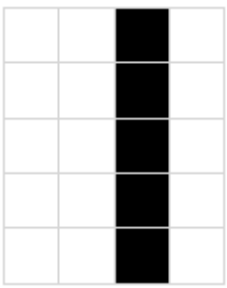
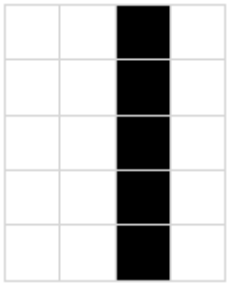
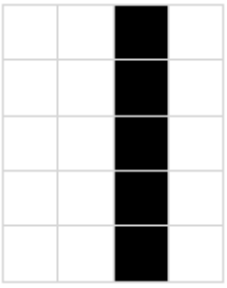
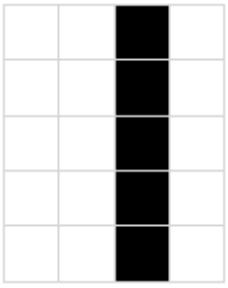
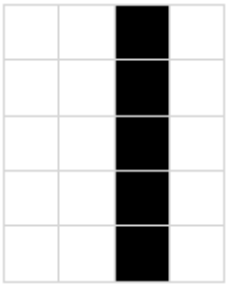
	Page: B0011			
Data	B0000	B0001	B0010	B0011
Pattern				
Data	B0100	B0101	B0110	B0111
Pattern				
Data	B1000	B1001	B1010	B1011
Pattern				
Data	B1100	B1101	B1110	B1111
Pattern				

	Page: B0100			
Data	B0000	B0001	B0010	B0011
Pattern				
Data	B0100	B0101	B0110	B0111
Pattern				
Data	B1000	B1001	B1010	B1011
Pattern				
Data	B1100	B1101	B1110	B1111
Pattern				

	Page: B0101			
Data	B0000	B0001	B0010	B0011
Pattern				
Data	B0100	B0101	B0110	B0111
Pattern				
Data	B1000	B1001	B1010	B1011
Pattern				
Data	B1100	B1101	B1110	B1111
Pattern				

	Page: B0110			
Data	B0000	B0001	B0010	B0011
Pattern				
Data	B0100	B0101	B0110	B0111
Pattern				
Data	B1000	B1001	B1010	B1011
Pattern				
Data	B1100	B1101	B1110	B1111
Pattern				

	Page: B0111			
Data	B0000	B0001	B0010	B0011
Pattern				
Data	B0100	B0101	B0110	B0111
Pattern				
Data	B1000	B1001	B1010	B1011
Pattern				
Data	B1100	B1101	B1110	B1111
Pattern				

	Page: B1000			
Data	B0000	B0001	B0010	B0011
Pattern				
Data	B0100	B0101	B0110	B0111
Pattern				
Data	B1000	B1001	B1010	B1011
Pattern				
Data	B1100	B1101	B1110	B1111
Pattern				

Page B1111 contains a self-diagnostic program, which tests every pixel, row and column individually.

Appendix B: Graphics Programming Table Template

You can photocopy this table and use it to design and document your own graphic programs for the B4.

Name of the Program _____

Author(s): _____

	Page RAM				Character RAM				Description
Step #	**3**	**2**	**1**	**0**	**3**	**2**	**1**	**0**	
Step 15									
Step 14									
Step 13									
Step 12									
Step 11									
Step 10									
Step 9									
Step 8									
Step 7									
Step 6									
Step 5									
Step 4									
Step 3									
Step 2									
Step 1									
Step 0									

Appendix C: Solutions

Here are the solutions to the tasks from the different chapters in this book.

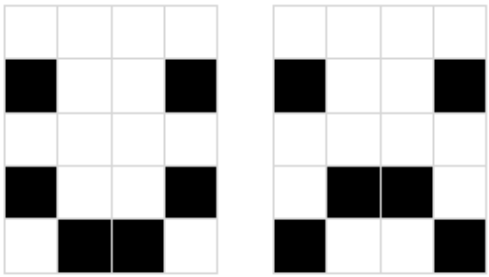
Question 3.1

Question: Using the method from this mission, output a word or sentence of your choice on the Dot Matrix Display. Use characters from at least 3 different pages.

Solution: This question has many solutions. Let's pick the output 'HAPPY :-)' Below is the resulting program:

	Page RAM				Character RAM				Description
Step #	**3**	**2**	**1**	**0**	**3**	**2**	**1**	**0**	
Step 15	0	0	0	1	1	0	1	0	Space
Step 14	0	0	0	1	1	0	1	0	Space
Step 13	0	0	0	1	1	0	1	0	Space
Step 12	0	0	0	1	1	0	1	0	Space
Step 11	0	0	0	1	1	0	1	0	Space
Step 10	0	0	0	1	1	0	1	0	Space
Step 9	0	0	0	1	1	0	1	0	Space
Step 8	0	0	0	1	1	0	1	0	Space
Step 7	0	0	0	1	1	0	1	0	Space
Step 6	0	0	0	1	1	0	1	0	Space
Step 5	0	0	1	1	0	0	1	1	Smiley
Step 4	0	0	0	1	1	0	0	0	Y
Step 3	0	0	0	0	1	1	1	1	P
Step 2	0	0	0	0	1	1	1	1	P
Step 1	0	0	0	0	0	0	0	1	A
Step 0	0	0	0	0	0	1	1	1	H

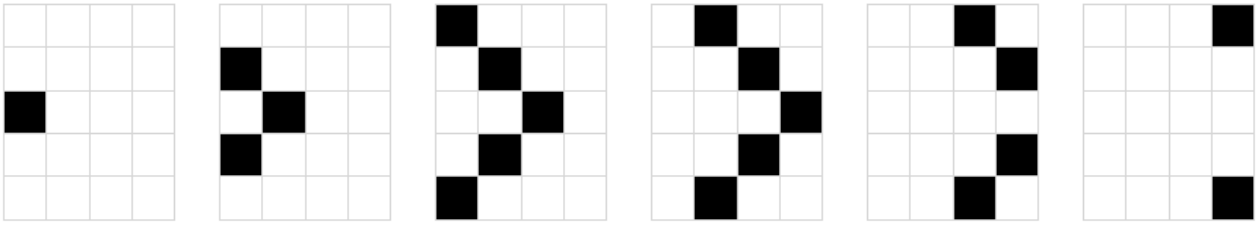
Question: Can you produce an animation of a smiley that goes from happy to sad?
Solution: We find the smiley faces on page B0011.



	Page RAM				Character RAM				Description
Step #	**3**	**2**	**1**	**0**	**3**	**2**	**1**	**0**	
Step 15	0	0	0	1	1	0	1	0	Space
Step 14	0	0	0	1	1	0	1	0	Space
Step 13	0	0	0	1	1	0	1	0	Space
Step 12	0	0	0	1	1	0	1	0	Space
Step 11	0	0	0	1	1	0	1	0	Space
Step 10	0	0	0	1	1	0	1	0	Space
Step 9	0	0	0	1	1	0	1	0	Space
Step 8	0	0	0	1	1	0	1	0	Space
Step 7	0	0	0	1	1	0	1	0	Space
Step 6	0	0	0	1	1	0	1	0	Space
Step 5	0	0	0	1	1	0	1	0	Space
Step 4	0	0	0	1	1	0	1	0	Space
Step 3	0	0	0	1	1	0	1	0	Space
Step 2	0	0	0	1	1	0	1	0	Space
Step 1	0	0	1	1	0	1	0	0	Sad Smiley
Step 0	0	0	1	1	0	0	1	1	Happy Smiley

Question: Can you produce an animation of an arrow that flies through the display from left to right and back?

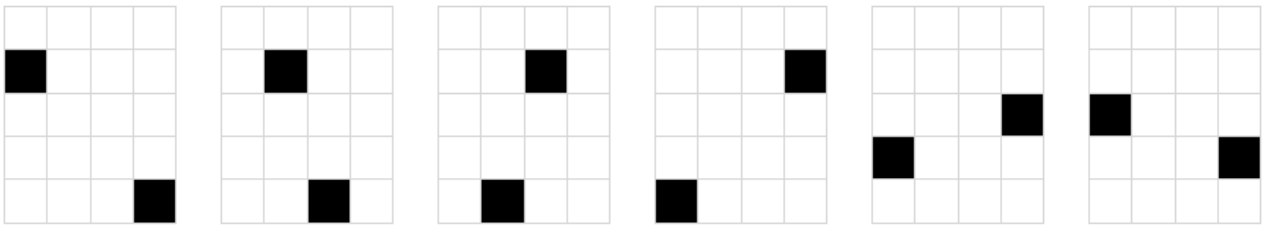
Solution: We find the necessary characters on page B0100. The animation consists of eleven steps. We run from 0 to six and then back to 0, giving the illusion of a moving arrow.



	Page RAM				Character RAM				Description
Step #	**3**	**2**	**1**	**0**	**3**	**2**	**1**	**0**	
Step 15	0	0	0	1	1	0	1	0	Space
Step 14	0	0	0	1	1	0	1	0	Space
Step 13	0	0	0	1	1	0	1	0	Space
Step 12	0	0	0	1	1	0	1	0	Space
Step 11	0	0	0	1	1	0	1	0	Space
Step 10	0	1	0	0	0	0	0	0	Arrow, part 0
Step 9	0	1	0	0	0	0	0	1	Arrow, part 1
Step 8	0	1	0	0	0	0	1	0	Arrow, part 2
Step 7	0	1	0	0	0	0	1	1	Arrow, part 3
Step 6	0	1	0	0	0	1	0	0	Arrow, part 4
Step 5	0	1	0	0	0	1	0	1	Arrow, part 5
Step 4	0	1	0	0	0	1	0	0	Arrow, part 4
Step 3	0	1	0	0	0	0	1	1	Arrow, part 3
Step 2	0	1	0	0	0	0	1	0	Arrow, part 2
Step 1	0	1	0	0	0	0	0	1	Arrow, part 1
Step 0	0	1	0	0	0	0	0	0	Arrow, part 0

Question: Can you produce an animation of two dots that circle around each other?

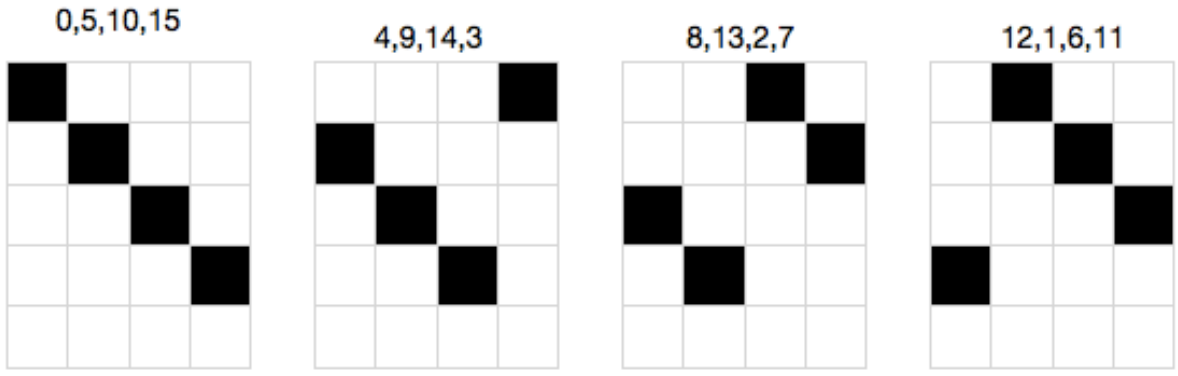
Solution: We find the necessary characters on page B0101. The animation consists of 6 images, so we have room for 2 animation sequences and fill the rest with empty space.



	Page RAM				Character RAM				Description
Step #	**3**	**2**	**1**	**0**	**3**	**2**	**1**	**0**	
Step 15	0	0	0	1	1	0	1	0	Space
Step 14	0	0	0	1	1	0	1	0	Space
Step 13	0	0	0	1	1	0	1	0	Space
Step 12	0	0	0	1	1	0	1	0	Space
Step 11	0	1	0	1	0	1	0	1	Dots, part 5
Step 10	0	1	0	1	0	1	0	0	Dots, part 4
Step 9	0	1	0	1	0	0	1	1	Dots, part 3
Step 8	0	1	0	1	0	0	1	0	Dots, part 2
Step 7	0	1	0	1	0	0	0	1	Dots, part 1
Step 6	0	1	0	1	0	0	0	0	Dots, part 0
Step 5	0	1	0	1	0	1	0	1	Dots, part 5
Step 4	0	1	0	1	0	1	0	0	Dots, part 4
Step 3	0	1	0	1	0	0	1	1	Dots, part 3
Step 2	0	1	0	1	0	0	1	0	Dots, part 2
Step 1	0	1	0	1	0	0	0	1	Dots, part 1
Step 0	0	1	0	1	0	0	0	0	Dots, part 0

Question 4.1		Solution
?	If you chose a different order for the pixels in your program. Would you still see the same picture?	The resulting picture would be the same.
	Draw a picture of your own choice and program it into the RAM modules. Execute the resulting program	There are many possible solutions. The design of a picture and the programming into the RAM modules is described in mission 4

Question 5.1		Solution
?	Why does the pattern not change when the bottom Variable is set to zero (B0000)?	Because zero will be added, leaving the output of the Adder unchanged a+0=a.
	Explain why the pattern seems to be moving down when the bottom Variable is set to B0001, but moving up when the value is B1111.	When 1 is added, the pixel right of the previous pixel lights up. Adding 15, however, is the same as subtracting 1. So, 2+15=17. In a 4 bit computer, that's 17-16=1. So by Adding B1111 the pixel left of the previous pixel lights up, giving the illusion of a moving up effect.
	Whilst the program is running, you can try to turn the bottom Variable and observe the results on the Dot Matrix Display. Try to produce different patters, observe similarities and try to explain what you see with what is happening inside the machine.	For example, by adding 5 we obtain the following sequence: 0,5,10,15,4,9,14,3,8,13,2,7,12,1,6,11,0. This looks like a diagonal line moving from the top right corner of the display towards the bottom left corner.



Appendix D: Quick Reference Guide

Page Number	Content
B0000	Letters A ... P
B0001	Letters Q ... Z
B0010	Numbers 0 ...15
B0011	Symbols, including smileys
B0100	Flying Arrow
B0101	Dancing Dots
B0110	Pixel Mode
B0111	Ones digit of input number
B1000	Tens digit of input number
B1111	Display diagnostics

Binary	Decimal
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	10
1011	11
1100	12
1101	13
1110	14
1111	15