



USB Flash / EPROM Programmer

<https://usbflashprog.robsonmartins.com>

Specifications

Version 0.1 - Rev. G

Author: Robson Martins (<https://www.robsonmartins.com>)

History

Revision	Date	Changes
G	Dec. 19, 2023	Added device algorithm: SRAM.
	Nov. 17, 2023	Removed "run by script" requirement. Removed the external power source.
F	May. 30, 2022	Added serial communication protocol (opcodes).
E	Apr. 23, 2022	CPU changed to Raspberry Pi Pico. Updated diagrams. Removed PT_BR translation.
D	Feb. 02, 2011	Added firmware project.
C	Dec. 27, 2010	Added adapter connectors pin-out.
B	Mar. 03, 2010	New Block Diagram.
A	Jan. 28, 2010	Initial Version.

Contents

1. Introduction.....	4
2. Requirements.....	4
3. Hardware Platform.....	5
3.1. Block Diagram.....	5
3.2. Functional Description.....	6
3.2.1. Main CPU.....	6
3.2.2. Power Supplies.....	6
3.2.3. Programmer Busses.....	6
3.2.4. Busses Interfaces.....	7
3.3. Adapter Connector Pin-Out.....	7
3.3.1. Parallel Adapter Connector.....	7
3.3.2. Serial Adapter Connector.....	9
4. Firmware Project.....	10
4.1. Block Diagram.....	10
4.2. Communication Protocol (Opcodes).....	11
4.2.1. Response Codes.....	11
4.2.2. Command Codes.....	11
5. Software Project.....	14
5.1. Device Algorithms.....	14
5.1.1. Parallel Memory – SRAM.....	14
Appendix A – Development Environment.....	20

1. Introduction

The purpose of this board is to allow the programming, reading and verification of writable/rewritable memory devices, such as EPROM, EEPROM, Flash, SRAM, NVRAM – those with parallel bus as well as serial ones (I2C, SPI, Microwire, LPC).

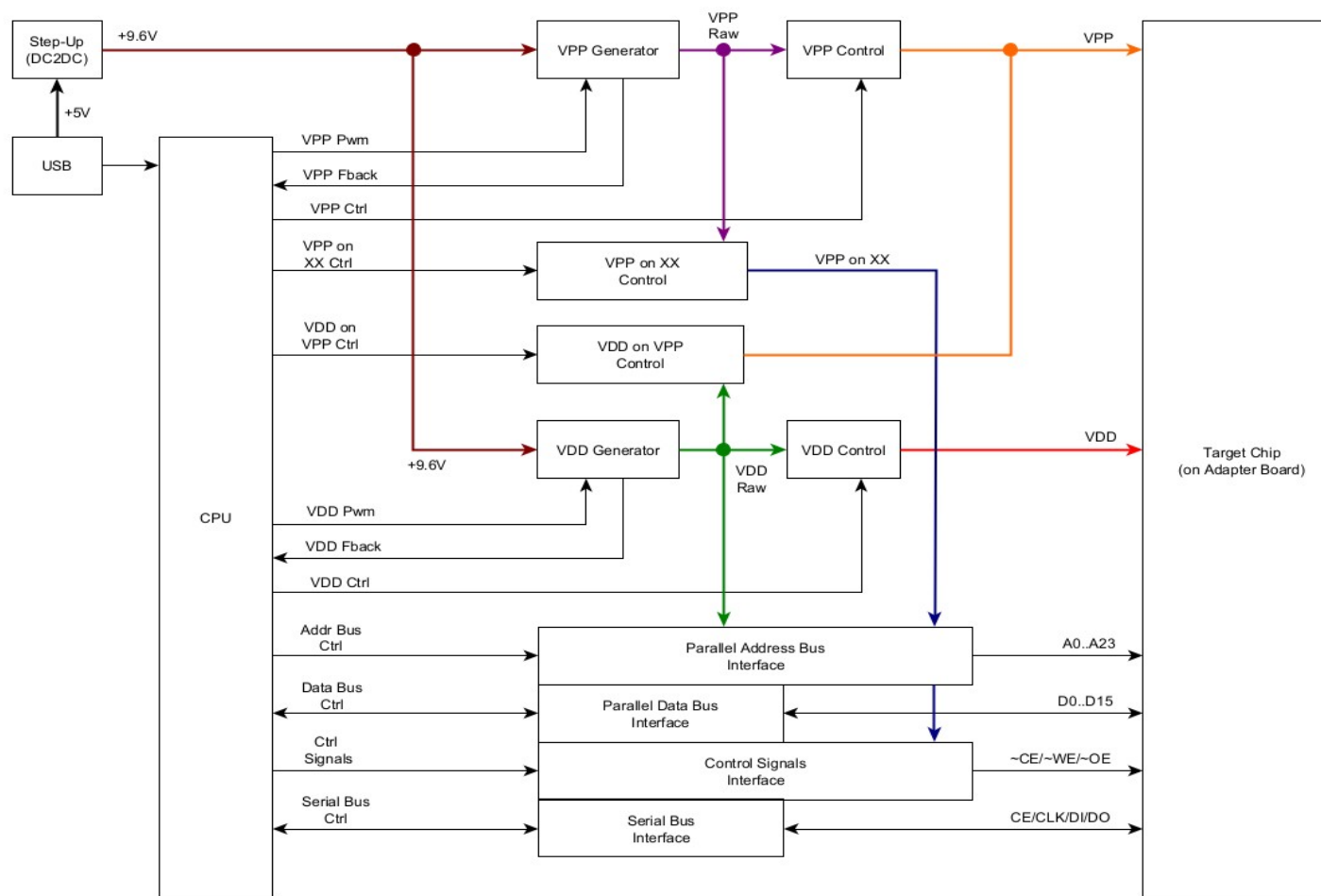
In a future release, programming of some microcontroller families (eg. Microchip PIC, or 8051) may also be supported, via firmware and software upgrade.

2. Requirements

- Allow write, read, delete, get ID and information about supported chips.
- Support parallel and serial devices (no microcontrollers in initial version).
- Support SRAM, EPROM, E2PROM, Flash, NVRAM, Hub/LPC devices (parallel and serial, including Microwire, I2C, SPI).
- Provide two sources of programming voltage: VDD (low voltage) and VPP/VEE (high voltage for write/erase), in the range between 3.3 V and 6.8 V (VDD), and between 12V and 25V (VPP/VEE).
- Automatic control for VDD and VPP/VEE voltages, according to the chip to be programmed.
- Allow *jumperless* chip configuration (by software chip selection).
- Socket for adapters – for each package or family of supported chips (no ZIF socket on the programmer board).
- Connection with PC via USB port, using a specific software for communication.
- Multi-platform software, compatible with Microsoft Windows® or GNU/Linux® operating systems, under 32 or 64 bits (if possible, Apple macOS® and FreeBSD versions can be available).
- Some compatibility with existing programmers adapters:
 - EzoFlash+ (<http://www.ezoflash.com/>).
 - MPSP (<https://mpsp.robsonmartins.com>).

3. Hardware Platform

3.1. Block Diagram



USB Flash/EPROM Programmer

Block Diagram

3.2. Functional Description

3.2.1. Main CPU

The CPU used in this programmer will be a Raspberry Pi Pico Module (with a RP2040 processor and USB support). This module has a dual core ARM Cortex-M0+ running at 133MHz, 256KB of SRAM and 2MB of storage, a USB port, required for communication between the programmer and the PC, plus one A/D converter with 3 inputs and 16 PWM channels (that can be used to generate the programming voltages). Moreover, there is a serial communication port (SPI / Microwire / I2C) that can be used for programming of serial devices, and GPIO pins to control parallel bus and signals.

3.2.2. Power Supplies

To generate the programming voltages (VDD and VPP/VEE), the programmer must have two DC/DC converters, driven by the PWM outputs of the CPU and monitored through of the ADC inputs.

The CPU will be powered directly by the voltage supplied by the USB port (5V). A fixed DC/DC converter will step up the voltage to 9.6V, to power the two DC/DC converters – VDD and VPP/VEE generators.

The CPU can turn on or off the VDD / VPP / VEE outputs, or supply VDD voltage on VPP line (via the “VDD on VPP” signal).

3.2.3. Programmer Busses

For handle parallel devices, the programmer must provide the following busses and signals to the target chip:

- **Address Bus (A0..A23)** – A addressing bus with 24 bits wide, allowing access up to 16777216 positions (16M).
- **Data Bus (D0..D7 / D8..D15)** – A data bus with 8 or 16 bits wide, allowing access for one byte (8 bits) or one word (16 bits), according to the memory width.
- **Control Lines (~CE / ~WE / ~OE)** – Chip Enable, Write Enable e Output Enable.
- **Power and Programming Voltages (VDD / VPP / VEE)** - Voltages used to power-up (VDD), program (VPP) or erase (VEE) the memory.

For handle serial devices, the programmer must provide the following signals to the target chip:

- **Clock (CLK)** – Clock line for synchronize the communication with the target memory.
- **Data Input (DIN)** – For read data from target memory.
- **Data Output (DOUT)** – For write data to target memory.

- **Control Lines (\sim CE / \sim WE / \sim OE)** – Chip Enable, Write Enable e Output Enable.
- **Power and Programming Voltages (VDD / VPP / VEE)** - Voltages used to power-up (VDD), program (VPP) or erase (VEE) the memory.

3.2.4. Busses Interfaces

To connect the microcontroller busses to target chip, is necessary adapt the voltage levels of the CPU (5V) and the voltage levels of the target chip ($3.3V \leq VDD \leq 6.8V$), using an interface circuitry.

3.3. Adapter Connector Pin-Out

3.3.1. Parallel Adapter Connector

Female (Top Side)

1	3	5	7	9	11	13	15	17	19	21	23	25	27	29	31	33	35	37	39	41	43	45	47
2	4	6	8	10	12	14	16	18	20	22	24	26	28	30	32	34	36	38	40	42	44	46	48

Pin	Function	Description
1	D0	DATA BUS – BIT 0
2	D1	DATA BUS – BIT 1
3	D2	DATA BUS – BIT 2
4	GND	GROUND
5	D3	DATA BUS – BIT 3
6	D4	DATA BUS – BIT 4
7	D5	DATA BUS – BIT 5
8	D6	DATA BUS – BIT 6
9	D7	DATA BUS – BIT 7
10	$\overline{\text{CE}}$	CHIP ENABLE
11	A10	ADDRESS BUS – BIT 10
12	$\overline{\text{OE}}$	OUTPUT ENABLE
13	A11	ADDRESS BUS – BIT 11
14	A9	ADDRESS BUS – BIT 9
15	A8	ADDRESS BUS – BIT 8
16	A13	ADDRESS BUS – BIT 13
17	A14	ADDRESS BUS – BIT 14
18	A17	ADDRESS BUS – BIT 17
19	$\overline{\text{WE}}$	WRITE ENABLE
20	VDD	VDD VOLTAGE

Pin	Function	Description
21	A18	ADDRESS BUS – BIT 18
22	A16	ADDRESS BUS – BIT 16
23	A15	ADDRESS BUS – BIT 15
24	A12	ADDRESS BUS – BIT 12
25	A7	ADDRESS BUS – BIT 7
26	A6	ADDRESS BUS – BIT 6
27	A5	ADDRESS BUS – BIT 5
28	A4	ADDRESS BUS – BIT 4
29	A3	ADDRESS BUS – BIT 3
30	A2	ADDRESS BUS – BIT 2
31	A1	ADDRESS BUS – BIT 1
32	A0	ADDRESS BUS – BIT 0
33	KEY	KEY TO AVOID CONNECTOR INVERSION
34	VPP	VPP PROGRAMMING VOLTAGE
35	A19	ADDRESS BUS – BIT 19
36	A20	ADDRESS BUS – BIT 20
37	A21	ADDRESS BUS – BIT 21
38	A22	ADDRESS BUS – BIT 22
39	A23	ADDRESS BUS – BIT 23
40	KEY	KEY TO AVOID CONNECTOR INVERSION
41	D8	DATA BUS – BIT 8
42	D9	DATA BUS – BIT 9
43	D10	DATA BUS – BIT 10
44	D11	DATA BUS – BIT 11
45	D12	DATA BUS – BIT 12
46	D13	DATA BUS – BIT 13
47	D14	DATA BUS – BIT 14
48	D15	DATA BUS – BIT 15

3.3.2. Serial Adapter Connector

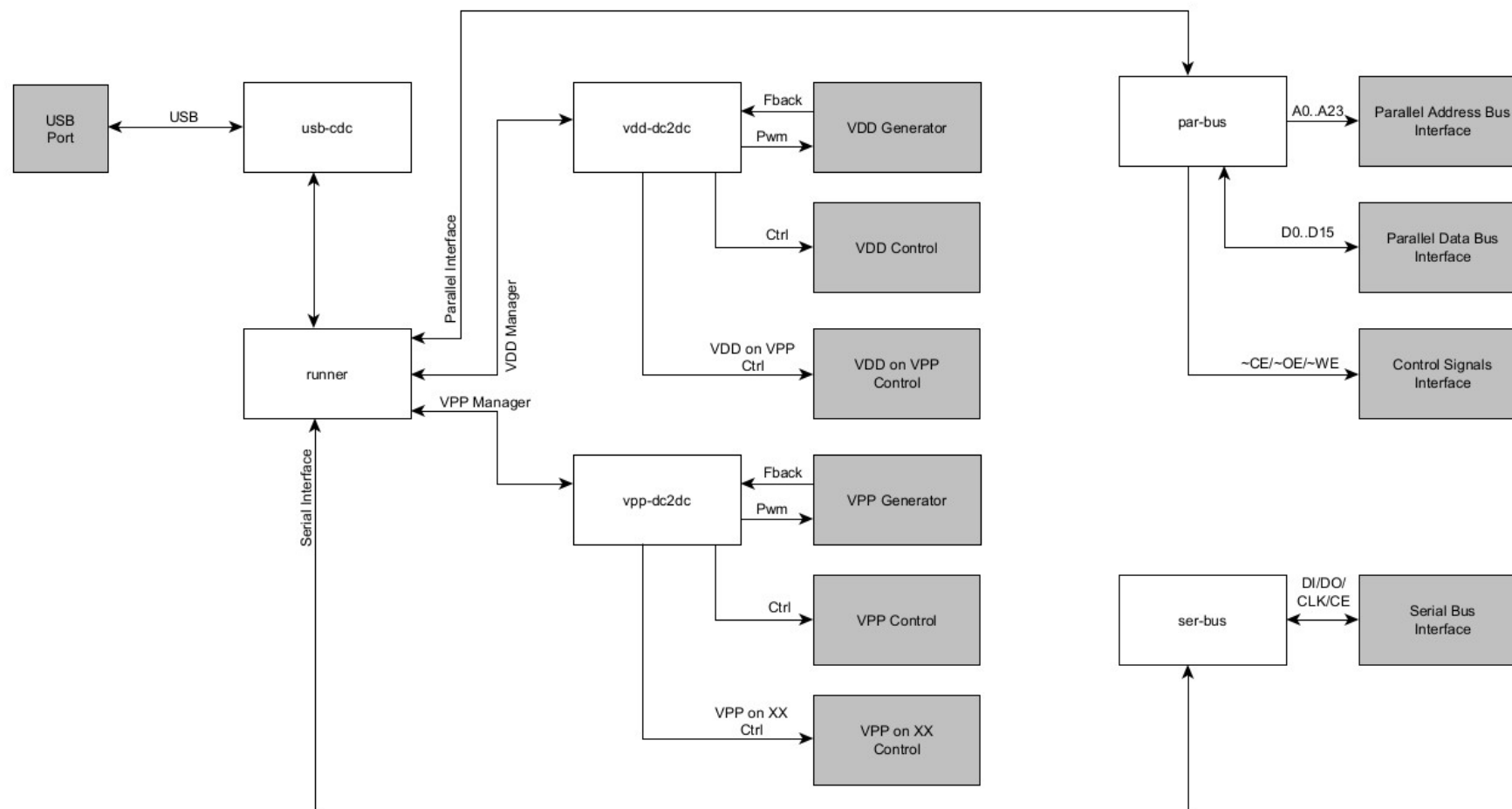
Female (Top Side)

1	3	5	7	9	11	13	15	17	19
2	4	6	8	10	12	14	16	18	20

Pin	Function	Description
1	VPP13	VPP PROGRAMMING VOLTAGE (13V)
2	VPP13	VPP PROGRAMMING VOLTAGE (13V)
3	VDD5	VDD VOLTAGE (5V)
4	VDD5	VDD VOLTAGE (5V)
5	GND	GROUND
6	GND	GROUND
7	SCK	SERIAL CLOCK
8	SCK	SERIAL CLOCK
9	GND	GROUND
10	GND	GROUND
11	SDO	SERIAL DATA OUT (TO TARGET SDI)
12	SDI	SERIAL DATA IN (FROM TARGET SDO)
13	GND	GROUND
14	GND	GROUND
15	VCC	VCC SUPPLY (5V – ALWAYS ON)
16	VCC	VCC SUPPLY (5V – ALWAYS ON)
17	\overline{WE}	CHIP SELECT (\overline{WE} PROGRAMMER PIN)
18	\overline{WE}	CHIP SELECT (\overline{WE} PROGRAMMER PIN)
19	GND	GROUND
20	KEY	KEY TO AVOID CONNECTOR INVERSION

4. Firmware Project

4.1. Block Diagram



USB Flash/EPROM Programmer

Firmware Block Diagram

4.2. Communication Protocol (Opcodes)

For communication between the programmer's hardware and the PC, the following protocol must be used (via USB-CDC serial class).

4.2.1. Response Codes

Op Code	Mnemonic	Description
0x00	CMD RESPONSE NOK	Response NOK.
0x01	CMD RESPONSE OK	Response OK.

4.2.2. Command Codes

Op Code	Mnemonic	Parameters	Response	Description
0x00	NOP	<none>	OK	No Operation.
0x01	CMD VDD CTRL	STATE	OK NOK	Set VDD Ctrl Pin On or Off. Parameter STATE is one byte size. If STATE == 0x00, will be OFF; If STATE != 0x00, will be ON.
0x02	CMD VDD SETV	VALUE	OK NOK	Set VDD Voltage. Parameter VALUE is two byte size. FIRST = Integer part of value SECOND = Fractional part of value
0x03	CMD VDD GETV	<none>	OK + VALUE NOK	Get VDD Voltage. Response VALUE is two byte size. FIRST = Integer part of value SECOND = Fractional part of value
0x04	CMD VDD GET%	<none>	OK + VALUE NOK	Get VDD PWM Duty Cycle (%). Response VALUE is two byte size. FIRST = Integer part of value SECOND = Fractional part of value
0x05	CMD VDD GETCAL	<none>	OK + VALUE NOK	Get VDD Calibration Value. Response VALUE is two byte size. FIRST = Integer part of value SECOND = Fractional part of value
0x06	CMD VDD INITCAL	<none>	OK NOK	Init VDD Calibration Process.
0x07	CMD VDD SAVECAL	VALUE	OK NOK	Save VDD Calibration Value. Parameter VALUE is two byte size. FIRST = Integer part of value SECOND = Fractional part of value
0x08	CMD VDD ON VPP	STATE	OK NOK	Set VDD On VPP Pin On or Off. Parameter STATE is one byte size. If STATE == 0x00, will be OFF; If STATE != 0x00, will be ON.

Op Code	Mnemonic	Parameters	Response	Description
0x11	CMD VPP CTRL	STATE	OK NOK	Set VPP Ctrl Pin On or Off. Parameter STATE is one byte size. If STATE == 0x00, will be OFF; If STATE != 0x00, will be ON.
0x12	CMD VPP SETV	VALUE	OK NOK	Set VPP Voltage. Parameter VALUE is two byte size. FIRST = Integer part of value SECOND = Fractional part of value
0x13	CMD VPP GETV	<none>	OK + VALUE NOK	Get VPP Voltage. Response VALUE is two byte size. FIRST = Integer part of value SECOND = Fractional part of value
0x14	CMD VPP GET%	<none>	OK + VALUE NOK	Get VPP PWM Duty Cycle (%). Response VALUE is two byte size. FIRST = Integer part of value SECOND = Fractional part of value
0x15	CMD VPP GETCAL	<none>	OK + VALUE NOK	Get VPP Calibration Value. Response VALUE is two byte size. FIRST = Integer part of value SECOND = Fractional part of value
0x16	CMD VPP INITCAL	<none>	OK NOK	Init VPP Calibration Process.
0x17	CMD VPP SAVECAL	VALUE	OK NOK	Save VPP Calibration Value. Parameter VALUE is two byte size. FIRST = Integer part of value SECOND = Fractional part of value
0x18	CMD VPP ON A9	STATE	OK NOK	Set VPP On A9 Pin On or Off. Parameter STATE is one byte size. If STATE == 0x00, will be OFF; If STATE != 0x00, will be ON.
0x19	CMD VPP ON A18	STATE	OK NOK	Set VPP On A18 Pin On or Off. Parameter STATE is one byte size. If STATE == 0x00, will be OFF; If STATE != 0x00, will be ON.
0x1A	CMD VPP ON CE	STATE	OK NOK	Set VPP On CE Pin On or Off. Parameter STATE is one byte size. If STATE == 0x00, will be OFF; If STATE != 0x00, will be ON.
0x1B	CMD VPP ON OE	STATE	OK NOK	Set VPP On OE Pin On or Off. Parameter STATE is one byte size. If STATE == 0x00, will be OFF; If STATE != 0x00, will be ON.
0x1C	CMD VPP ON WE	STATE	OK NOK	Set VPP On WE Pin On or Off. Parameter STATE is one byte size. If STATE == 0x00, will be OFF; If STATE != 0x00, will be ON.
0x21	CMD BUS CE CTRL	STATE	OK NOK	Set CE Pin On or Off. Parameter STATE is one byte size. If STATE == 0x00, will be OFF; If STATE != 0x00, will be ON.

Op Code	Mnemonic	Parameters	Response	Description
0x22	CMD BUS OE CTRL	STATE	OK NOK	Set OE Pin On or Off. Parameter STATE is one byte size. If STATE == 0x00, will be OFF; If STATE != 0x00, will be ON.
0x23	CMD BUS WE CTRL	STATE	OK NOK	Set WE Pin On or Off. Parameter STATE is one byte size. If STATE == 0x00, will be OFF; If STATE != 0x00, will be ON.
0x31	CMD BUS AD CLR	<none>	OK NOK	Clear Address Bus Value (Set Address to 0x00).
0x32	CMD BUS AD INC	<none>	OK NOK	Increment Address Bus Value.
0x33	CMD BUS AD SET	VALUE	OK NOK	Set Address Bus Value. Parameter VALUE is three byte size. FIRST = HI SECOND = MID THIRD = LOW
0x34	CMD BUS AD SETB	VALUE	OK NOK	Set Address Bus Value (BYTE). Parameter VALUE is one byte size. VALUE = LOW
0x35	CMD BUS AD SETW	VALUE	OK NOK	Set Address Bus Value (WORD). Parameter VALUE is two byte size. FIRST = MID SECOND = LOW
0x41	CMD BUS DT CLR	<none>	OK NOK	Clear Data Bus Value (Set Data to 0x00).
0x42	CMD BUS DT SET	VALUE	OK NOK	Set Data Bus Value. Parameter VALUE is two byte size. FIRST = HI SECOND = LOW
0x43	CMD BUS DT SETB	VALUE	OK NOK	Set Data Bus Value (BYTE). Parameter VALUE is one byte size. VALUE = LOW
0x44	CMD BUS DT GET	<none>	OK + VALUE NOK	Get Data Bus Value. Response VALUE is two byte size. FIRST = HI SECOND = LOW
0x45	CMD BUS DT GETB	<none>	OK + VALUE NOK	Get Data Bus Value (BYTE). Response VALUE is one byte size. VALUE = LOW
TODO	CMD SERIAL BUS			TODO: Serial Bus Commands.

5. Software Project

5.1. Device Algorithms

To read, write and erase memory devices, some algorithms are defined in the software. These algorithms are described below.

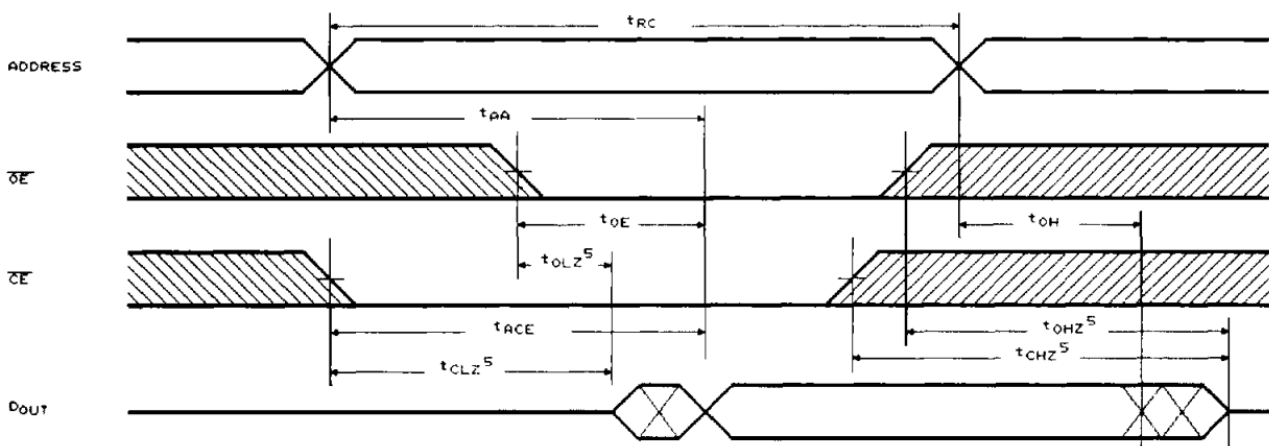
5.1.1. Parallel Memory – SRAM

Static Random-Access Memory (SRAM) is a type of random-access memory (RAM) that uses latching circuitry (flip-flop) to store each bit. SRAM is volatile memory; data is lost when power is removed.

The most common SRAM chips operate on 5V power supply (VDD), and the same voltage is used for write and erase (VPP).

The SRAM read cycle can be illustrated:

Read Cycle 1⁽¹⁾

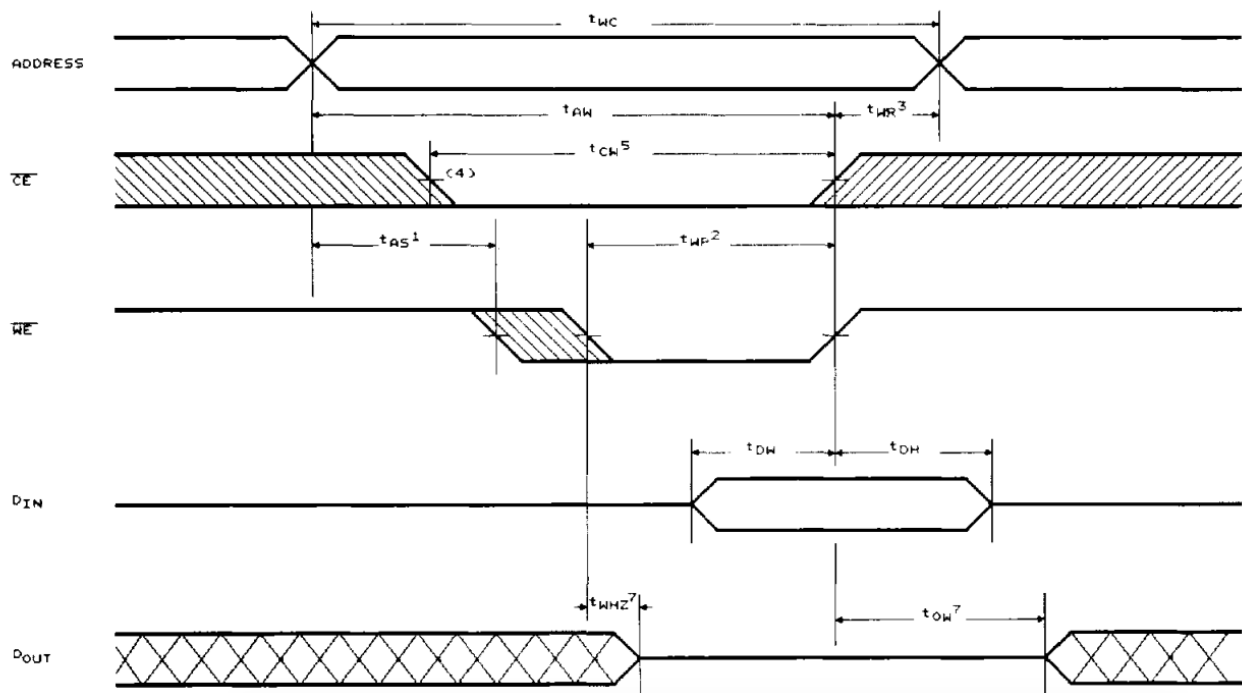


To read a SRAM, the following steps are required:

1. Power the SRAM with VDD;
2. Put \overline{WE} , \overline{OE} in HI;
3. Put \overline{CE} in LO;
4. Put the address on bus A0..An;
5. Put \overline{OE} in LO;
6. The data will be available on bus D0..Dn.

The SRAM write cycle can be illustrated:

Write Cycle 1⁽⁶⁾
(Write Enable Controlled)

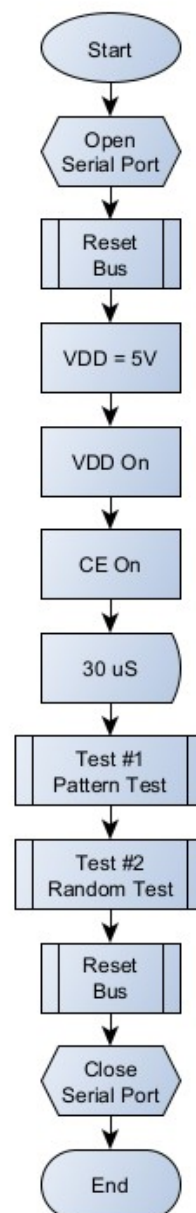


To write a SRAM, the following steps are required:

1. Power the SRAM with VDD;
2. Put \overline{WE} , \overline{OE} in HI;
3. Put \overline{CE} in LO;
4. Put the address on bus A0..An;
5. Put the data on bus D0..Dn;
6. Put \overline{WE} in LO;
7. Wait for t_{WP} time;
8. Put \overline{WE} in HI;
9. The data will be recorded in memory.

As SRAM is a volatile memory, it isn't possible to write for later reading, as data is lost when the power is turned off. So, instead, a SRAM memory testing algorithm was proposed, as follows.

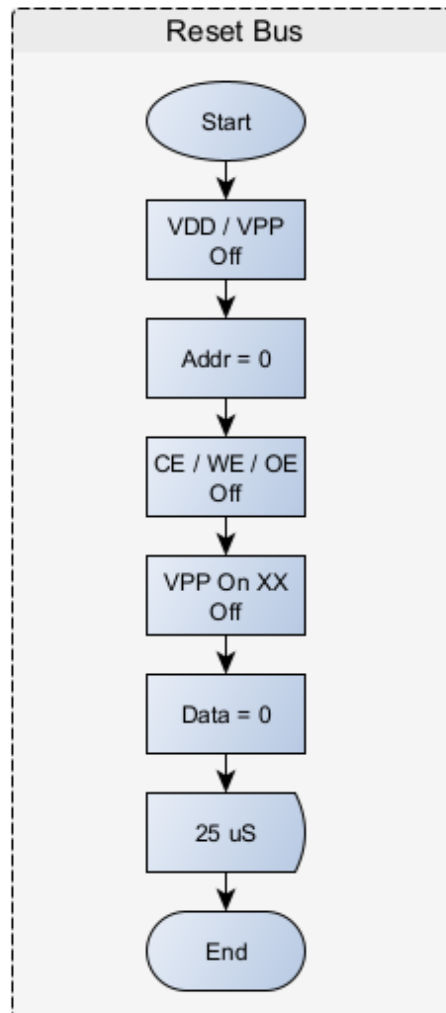
SRAM Test Algorithm



In this algorithm, the memory runs two tests: one is the pattern test, which writes the patterns 01010101 and 10101010 alternately to the memory addresses. The other is the random test, which writes a random number to each memory location. In both tests, the memory is written and read at each address, being checked, and then it is completely read, from beginning to end, being checked again against the recorded data.

Reset Bus Routine

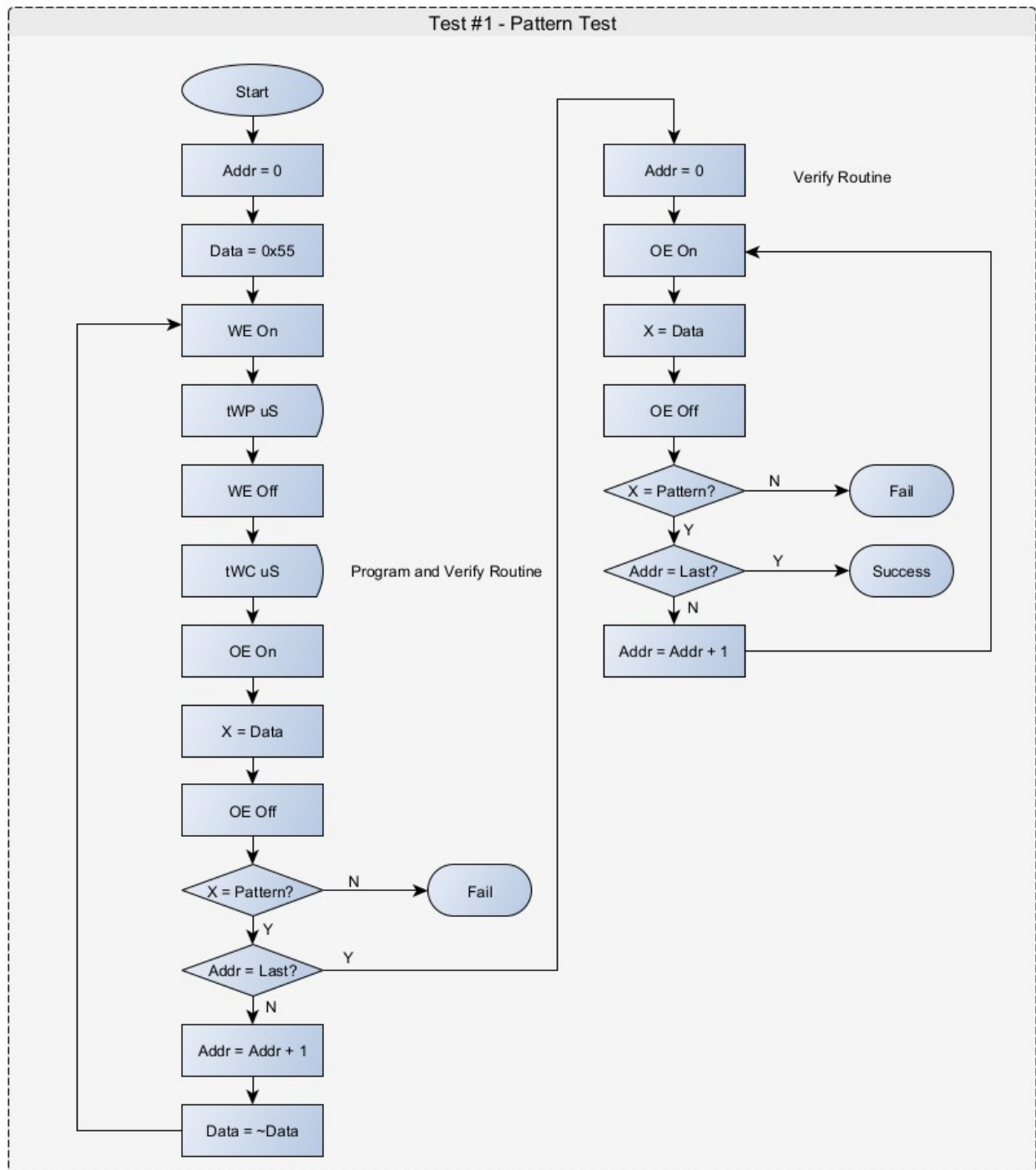
The Reset Bus routine is illustrated below.



This routine is responsible for initializing the buses and pins.

Test #1 Routine

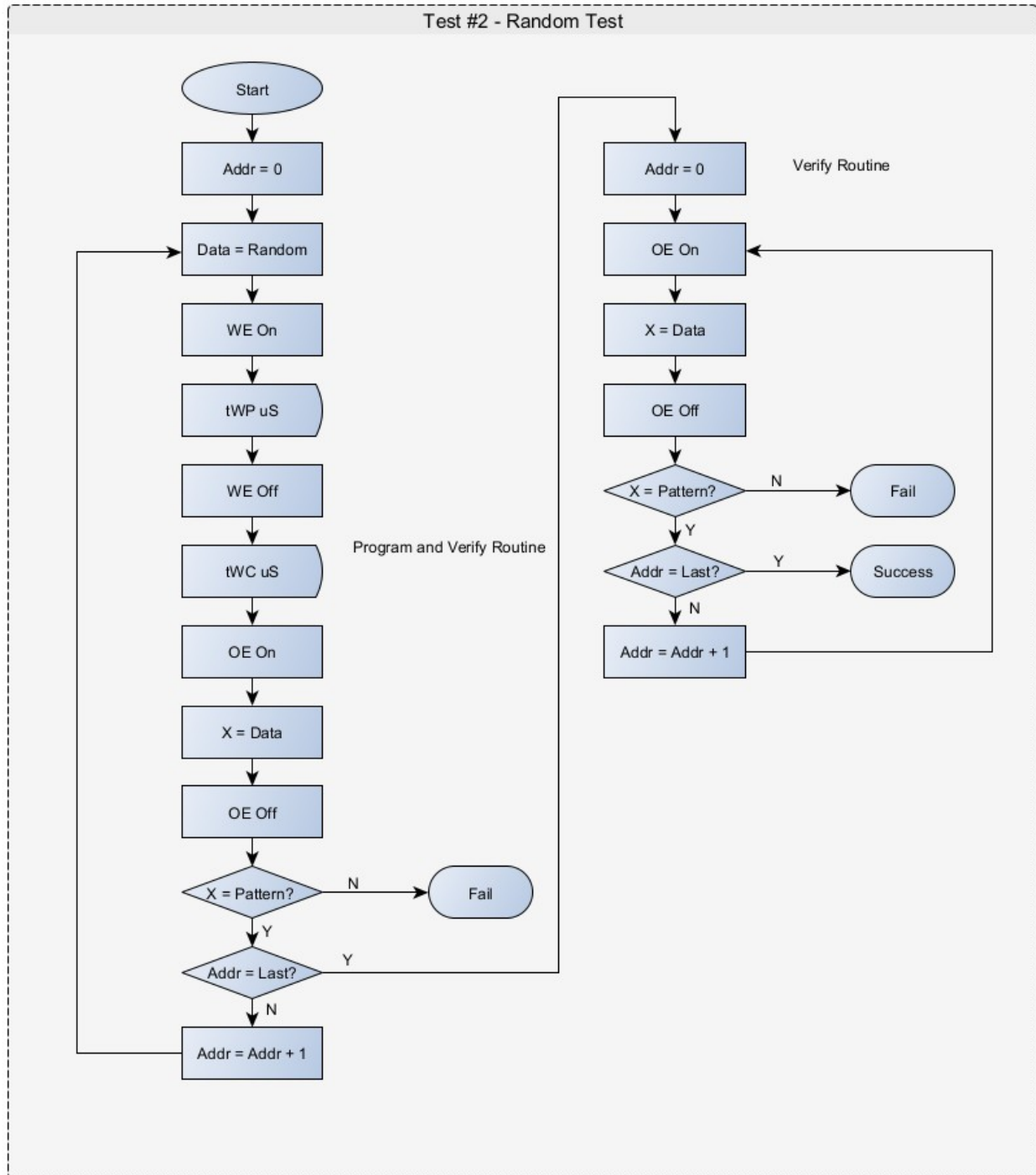
The Test #1 routine is illustrated below.



This routine is responsible for running the pattern test in memory.

Test #2 Routine

The Test #2 routine is illustrated below.



This routine is responsible for running the random test in memory.

Appendix A – Development Environment

To develop the programmer, should be used only open source and freeware software:

- **Operating System:**
 - GNU/Linux (<https://distrowatch.com/>)
- **Documentation:**
 - LibreOffice (<https://www.libreoffice.org/>)
 - yEd Graph Editor (<https://www.yworks.com/products/yed>)
- **Hardware Development:**
 - CAD:
 - Kicad (<https://www.kicad.org>)
- **Firmware Development:**
 - Raspberry Pi Pico Module:
 - Raspberry Pi Pico (<https://www.raspberrypi.com/products/raspberry-pi-pico/>)
- **Software Development:**
 - C/C++ Compiler:
 - GCC (<https://gcc.gnu.org/>)
 - GUI Framework:
 - Qt (<https://www.qt.io>)
 - IDE:
 - Qt Creator (<https://www.qt.io/product/development-tools>)
 - Microsoft Visual Studio Code (<https://code.visualstudio.com/>)
 - Code Documentation:
 - Doxygen (<https://www.doxygen.org/>)
- **Version Control System:**
 - Git (<https://git-scm.com/>)