USB Flash / EPROM Programmer

https://usbflashprog.robsonmartins.com

Specifications

Version 0.3

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

History

Revision	Date	Changes
0.3 A	Feb. 17, 2024	Updated device opcodes of serial communication protocol.
	Feb. 03, 2024	Updated device opcodes of serial communication protocol.
0.3	Jan. 31, 2024	Added device algorithm: EEPROM.
		Created new opcodes for device programming.
0.2	Jan. 24, 2024	Added pinout compatibility appendices.
	Jan. 17, 2024	Created new opcodes for device programming.
	Jan. 03, 2024	Added device algorithm: Electrically Erasable EPROM.
	Jan. 02, 2024	Added device algorithm: EPROM.
0.1 G	Dec. 19, 2023	Added device algorithm: SRAM.
	Nov. 17, 2023	Removed “run by script” requirement.
		Removed the external power source.
0.1 F	May. 30, 2022	Added serial communication protocol (opcodes).
0.1 E	Apr. 23, 2022	CPU changed to Raspberry Pi Pico.
		Updated diagrams.
		Removed PT_BR translation.
0.1 D	Feb. 02, 2011	Added firmware project.
0.1 C	Dec. 27, 2010	Added adapter connectors pin-out.
0.1 B	Mar. 03, 2010	New Block Diagram.
0.1 A	Jan. 28, 2010	Initial Version.

Contents

List of Figures

List of Tables

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

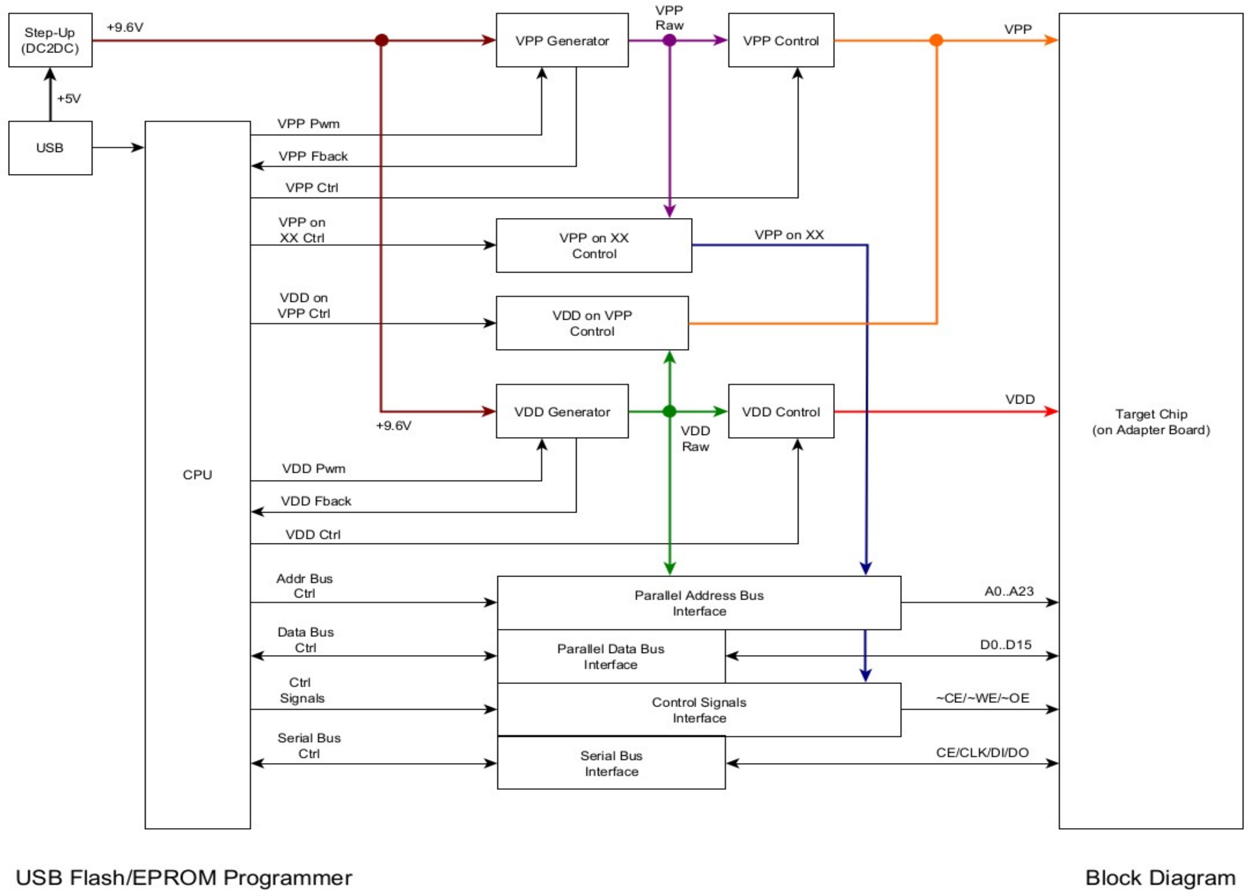


Figure 1: USB Flash/EPROM Programmer Block Diagram

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 (~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.

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 <= VDD <= 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

Table 1: Parallel Adapter Connector Pin-Out

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	$\overline{\text{CE}}$	$\overline{\text{CHIP ENABLE}}$
11	A10	ADDRESS BUS – BIT 10
12	$\overline{\text{OE}}$	$\overline{\text{OUTPUT ENABLE}}$
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	$\overline{\text{WE}}$	$\overline{\text{WRITE ENABLE}}$

Pin	Function	Description
20	VDD	VDD VOLTAGE
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

Table 2: Serial Adapter Connector Pin-Out

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

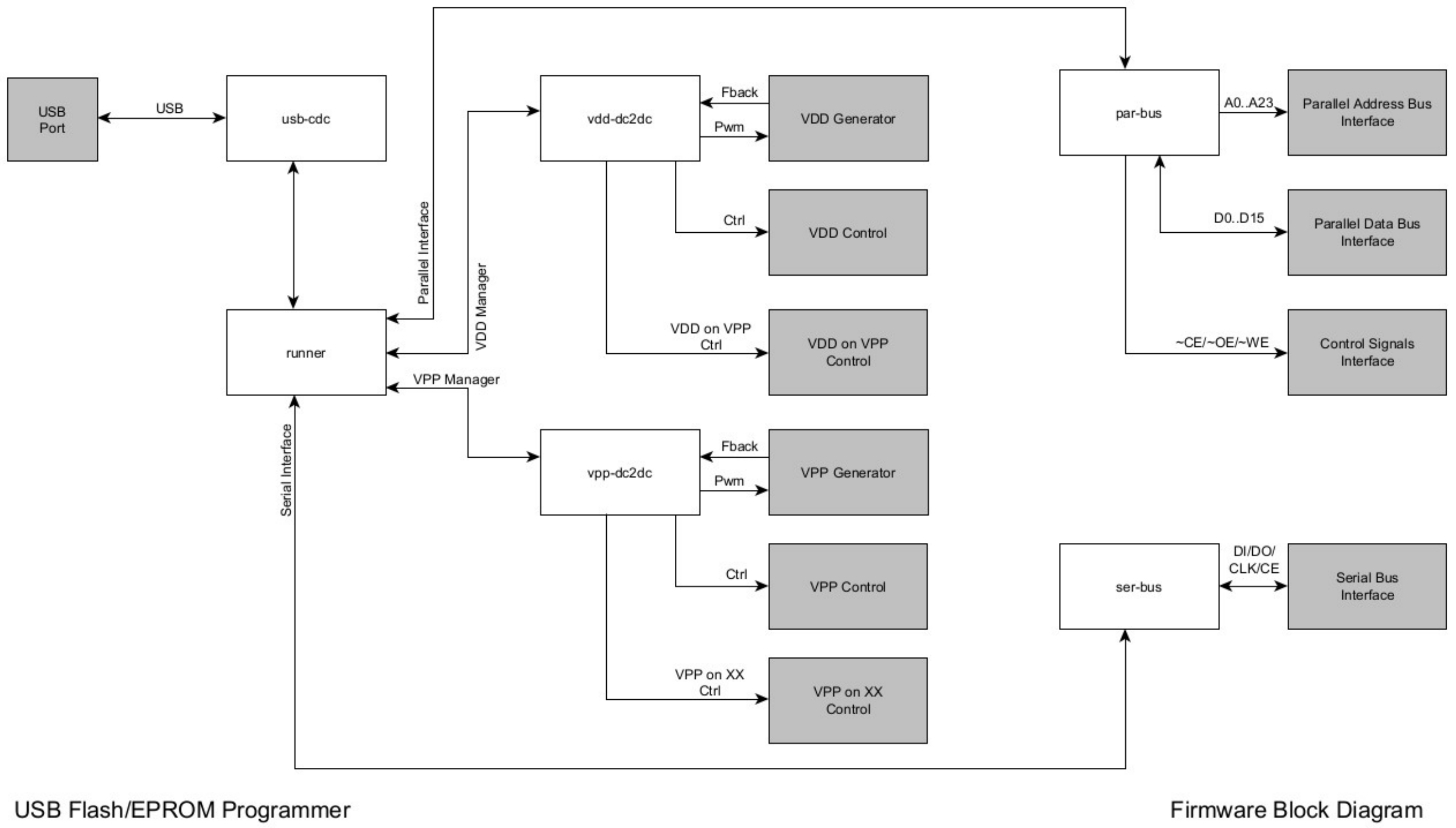


Figure 2: USB Flash/EPROM Programmer Firmware Block Diagram

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

Table 3: Communication Protocol – Response Codes

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

4.2.2. Command Codes

Table 4: Communication Protocol – Command Codes (Low Level)

Op Code	Mnemonic	Parameters	Response	Description
		Low Level Commands		
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

Op Code	Mnemonic	Parameters	Response	Description

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.`

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.`

Op Code	Mnemonic	Parameters	Response	Description

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.`

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 (DWORD).
				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 (BYTE). Parameter VALUE is one byte size. VALUE = LOW

0x43	CMD BUS DT SETW	VALUE	OK
			NOK
				Set Data Bus Value (WORD).
				Parameter VALUE is two byte size.
				FIRST = HI
				SECOND = LOW

0x44	CMD BUS DT GET	<none>	OK + VALUE NOK	Get Data Bus Value (BYTE). Response VALUE is one byte size. VALUE = LOW

0x45	CMD BUS DT GETW	<none>	OK + VALUE
			NOK
				Get Data Bus Value (WORD).
				Response VALUE is two byte size.
				FIRST = HI
				SECOND = LOW

TODO	CMD SERIAL BUS			TODO: Serial Bus Commands.

Table 5: Communication Protocol – Command Codes (High Level)

Op Code	Mnemonic	Parameters	Response	Description
High Level Commands				

0x81	CMD DEVICE SET TWP	VALUE	OK	Set the tWP Value for Programming a Device.
				Parameter VALUE is four byte size.
				FIRST = MSB
				FOURTH = LSB

0x82	CMD DEVICE SET TWC	VALUE	OK	Set the tWC Value for Programming a Device.
				Parameter VALUE is four byte size.
				FIRST = MSB
				FOURTH = LSB

0x83	CMD DEVICE CONFIGURE	VALUE	OK	Set the Device Flags for Programming a Device. Parameter VALUE is two byte size. FIRST = DEVICE ALGORITHM SECOND = FLAGS, as defined: **Bit** **Description** 0 Skip Write of 0xFF 1 Program with VPP on 2 VPP/$\overline{OE}$ pin 3 $\overline{PGM}$/$\overline{CE}$ pin 4 PGM is positive 5 16-bit mode
0x84	CMD DEVICE SETUP BUS	VALUE	OK NOK	Setup Busses (Data, Address and Control) to perform an operation on a Device. Parameter VALUE is one byte size. **Value** **Description** 0x00 Reset Bus 0x01 Prepare to Read 0x02 Prepare to Program Prerequisites: • CMD DEVICE CONFIGURE • CMD VDD SETV (Read) • CMD VPP SETV (Read and Program)

Op Code	Mnemonic	Parameters	Response	Description
0x85	CMD DEVICE READ	VALUE	OK + DATA NOK	Read Data Buffer (BYTE/WORD) from Device at current address, and Increment the address. Parameter VALUE is one byte size, and represents the number of bytes to be read (N). Response DATA is N byte size. for WORD: FIRST = HI; SECOND = LOW, in sequence. Prerequisites: • CMD DEVICE CONFIGURE • CMD VDD SETV • CMD DEVICE SETUP BUS (Read)
0x86	CMD DEVICE WRITE	VALUE + DATA	OK NOK	Program Data Buffer (BYTE/WORD) to Device at current address, and Increment the address. Parameter VALUE is one byte size, and represents the number of bytes to be write (N). Parameter DATA is N byte size. for WORD: FIRST = HI; SECOND = LOW, in sequence. Prerequisites: • CMD DEVICE CONFIGURE • CMD DEVICE SET TWP • CMD DEVICE SET TWC • CMD VDD SETV • CMD VPP SETV • CMD DEVICE SETUP BUS (Program)
0x87	CMD DEVICE WRITESECTOR	VALUE + DATA	OK NOK	Program Data Sector (BYTE/WORD) to Device at current address, and Increment the address. Parameter VALUE is two byte size, and represents the number of bytes to be write (SECTOR_SIZE). FIRST = HI SECOND = LOW Parameter DATA is SECTOR_SIZE byte size. for WORD: FIRST = HI; SECOND = LOW, in sequence. Prerequisites: • CMD DEVICE CONFIGURE • CMD DEVICE SET TWP • CMD DEVICE SET TWC • CMD VDD SETV • CMD VPP SETV • CMD DEVICE SETUP BUS (Program)

Op Code	Mnemonic	Parameters	Response	Description
0x88	CMD DEVICE VERIFY	VALUE + DATA	OK NOK	Verify Data Buffer (BYTE/WORD) from Device at current address, and Increment the address. Parameter VALUE is one byte size, and represents the number of bytes to be verify (N). Parameter DATA is N byte size. for WORD: FIRST = HI; SECOND = LOW, in sequence. Prerequisites: • CMD DEVICE CONFIGURE • CMD VDD SETV • CMD DEVICE SETUP BUS (Read)
0x89	CMD DEVICE BLANKCHECK	VALUE	OK NOK	Blank Check Data Buffer (BYTE/WORD) from Device at current address, and Increment the address. Parameter VALUE is one byte size, and represents the number of bytes to be check (N). Prerequisites: • CMD DEVICE CONFIGURE • CMD VDD SETV • CMD DEVICE SETUP BUS (Read)
0x8A	CMD DEVICE GET ID	<none>	OK + VALUE NOK	Get ID from Device. Response VALUE is four byte size. **Byte** **Description** First (MSB)/Second (LSB) Manufacurer ID Third (MSB)/Fourth (LSB) Device ID Prerequisites: • CMD DEVICE CONFIGURE • CMD VDD SETV • CMD VPP SETV • CMD DEVICE SETUP BUS (Read)
0x8B	CMD DEVICE ERASE	<none>	OK NOK	Erase All Data from Device. Prerequisites: • CMD DEVICE CONFIGURE • CMD DEVICE SET TWP • CMD DEVICE SET TWC • CMD VDD SETV • CMD VPP SETV • CMD DEVICE SETUP BUS (Program)

Byte	Description
First (MSB)/Second (LSB)	Manufacurer ID
Third (MSB)/Fourth (LSB)	Device ID

Op Code	Mnemonic	Parameters	Response	Description
0x8C	CMD DEVICE UNPROTECT	<none>	OK NOK	Unprotect entire Device for writing. Prerequisites: • CMD DEVICE CONFIGURE • CMD DEVICE SET TWP • CMD DEVICE SET TWC • CMD VDD SETV • CMD VPP SETV • CMD DEVICE SETUP BUS (Program)
0x8D	CMD DEVICE PROTECT	<none>	OK NOK	Protect entire Device for writing. Prerequisites: • CMD DEVICE CONFIGURE • CMD DEVICE SET TWP • CMD DEVICE SET TWC • CMD VDD SETV • CMD VPP SETV • CMD DEVICE SETUP BUS (Program)

Table 6: Device Algorithms

Value	Description
0x00	Algorithm Unknown
0x04	Algorithm SRAM
0x08	Algorithm EPROM
0x0C	Algorithm EEPROM (28C64)
0x0D	Algorithm EEPROM (28C256 or upper)
0x10	Algorithm Flash 28F
0x11	Algorithm Flash SST28SF
0x12	Algorithm Flash Am28F
0x13	Algorithm Flash i28F

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).

Common pinouts

Considering most SRAMs encapsulated with DIP packages, the most common pinouts are shown below:

Table 7: SRAM DIP24 – Pinout

xx16 (2K)		Chip		xx16 (2K)
A7	1		24	VDD
A6	2		23	A8
A5	3		22	A9
A4	4		21	$\overline{WE}$
A3	5		20	$\overline{OE}$
A2	6		19	A10
A1	7		18	$\overline{CE}$
A0	8		17	D7
D0	9		16	D6
D1	10		15	D5
D2	11		14	D4
GND	12		13	D3

Table 8: SRAM DIP28 – Pinout

xx256 (32K)	xx128 (16K)	xx64 (8K)		Chip		xx64 (8K)	xx128 (16K)	xx256 (32K)
A14	NC	NC	1		28	VDD	VDD	VDD
A12	A12	A12	2		27	$\overline{WE}$	$\overline{WE}$	$\overline{WE}$
A7	A7	A7	3		26	NC	A13	A13
A6	A6	A6	4		25	A8	A8	A8
A5	A5	A5	5		24	A9	A9	A9
A4	A4	A4	6		23	A11	A11	A11
A3	A3	A3	7		22	$\overline{OE}$	$\overline{OE}$	$\overline{OE}$
A2	A2	A2	8		21	A10	A10	A10
A1	A1	A1	9		20	$\overline{CE}$	$\overline{CE}$	$\overline{CE}$
A0	A0	A0	10		19	D7	D7	D7
D0	D0	D0	11		18	D6	D6	D6
D1	D1	D1	12		17	D5	D5	D5
D2	D2	D2	13		16	D4	D4	D4
GND	GND	GND	14		15	D3	D3	D3

Read Cycle

The SRAM read cycle can be illustrated:

Read Cycle 1 (1)

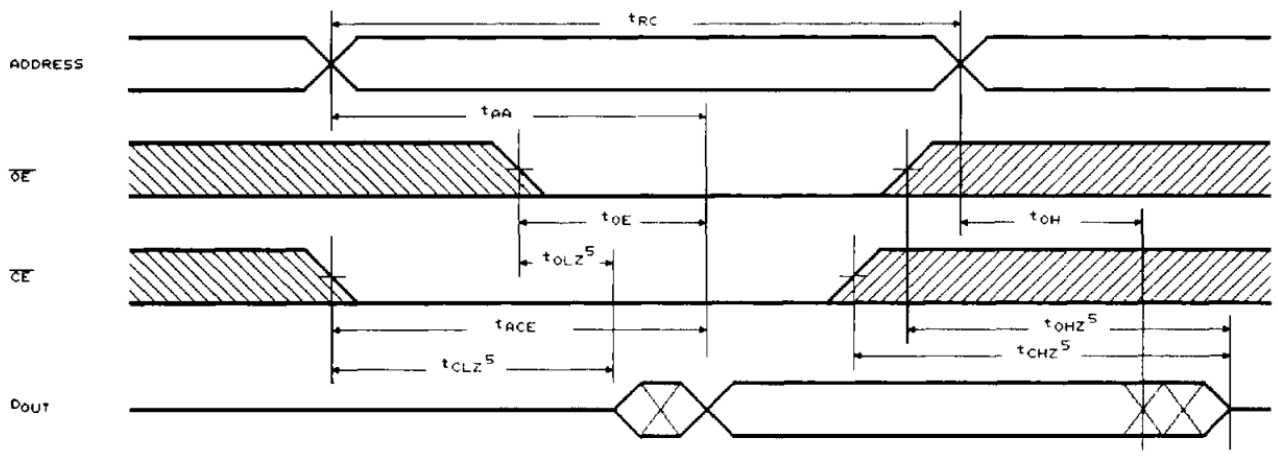


Figure 3: SRAM Read Cycle

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.

Write Cycle

The SRAM write cycle can be illustrated:

Write Cycle 1 (6)
(Write Enable Controlled)

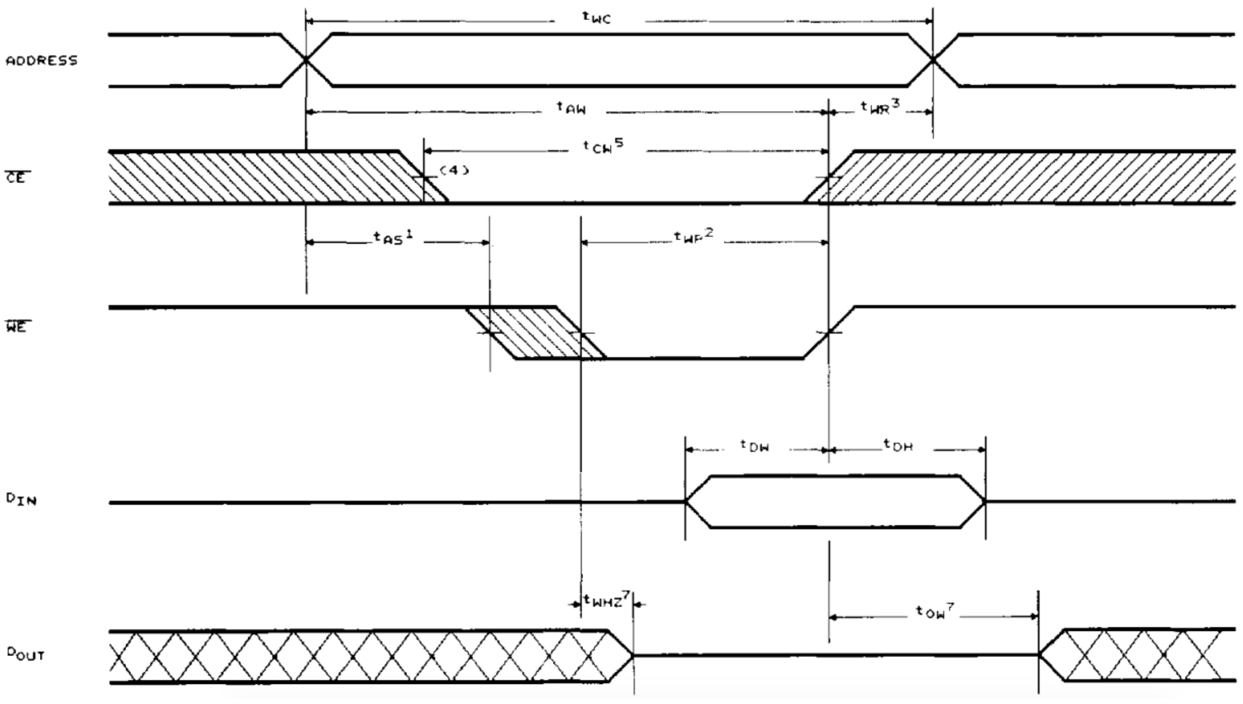


Figure 4: SRAM Write Cycle

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 tWP time;
8. Put $\overline{WE}$ in HI;
9. The data will be recorded in memory.

SRAM Test Algorithm

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.

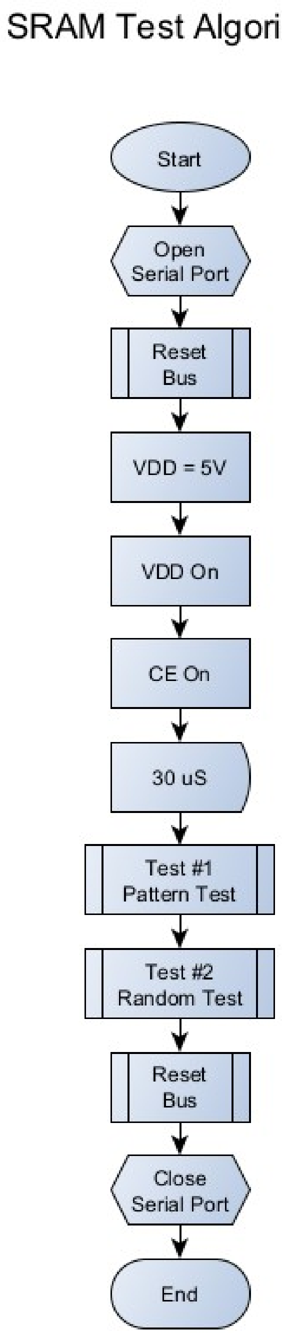


Figure 5: SRAM Test Algorithm

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.

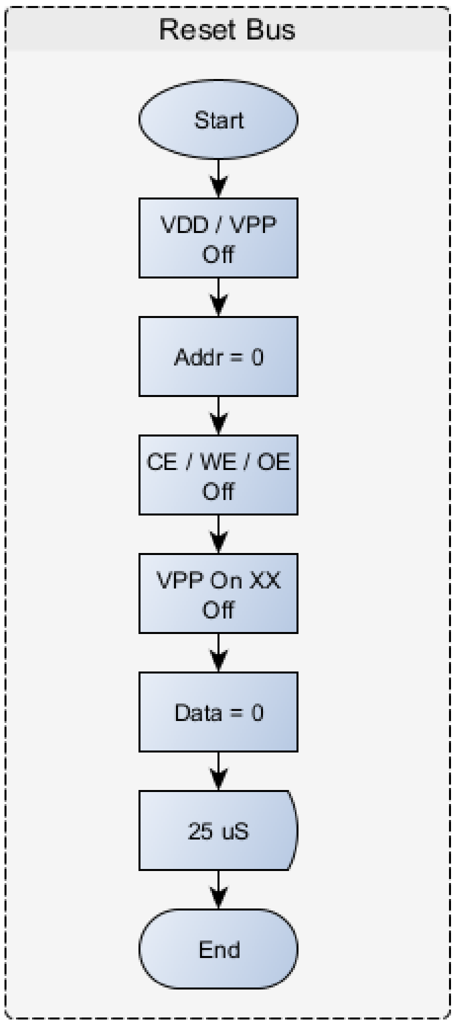


Figure 6: SRAM Reset Bus Routine

This routine is responsible for initializing the buses and pins.

Test #1 Routine

The Test #1 routine is illustrated below.

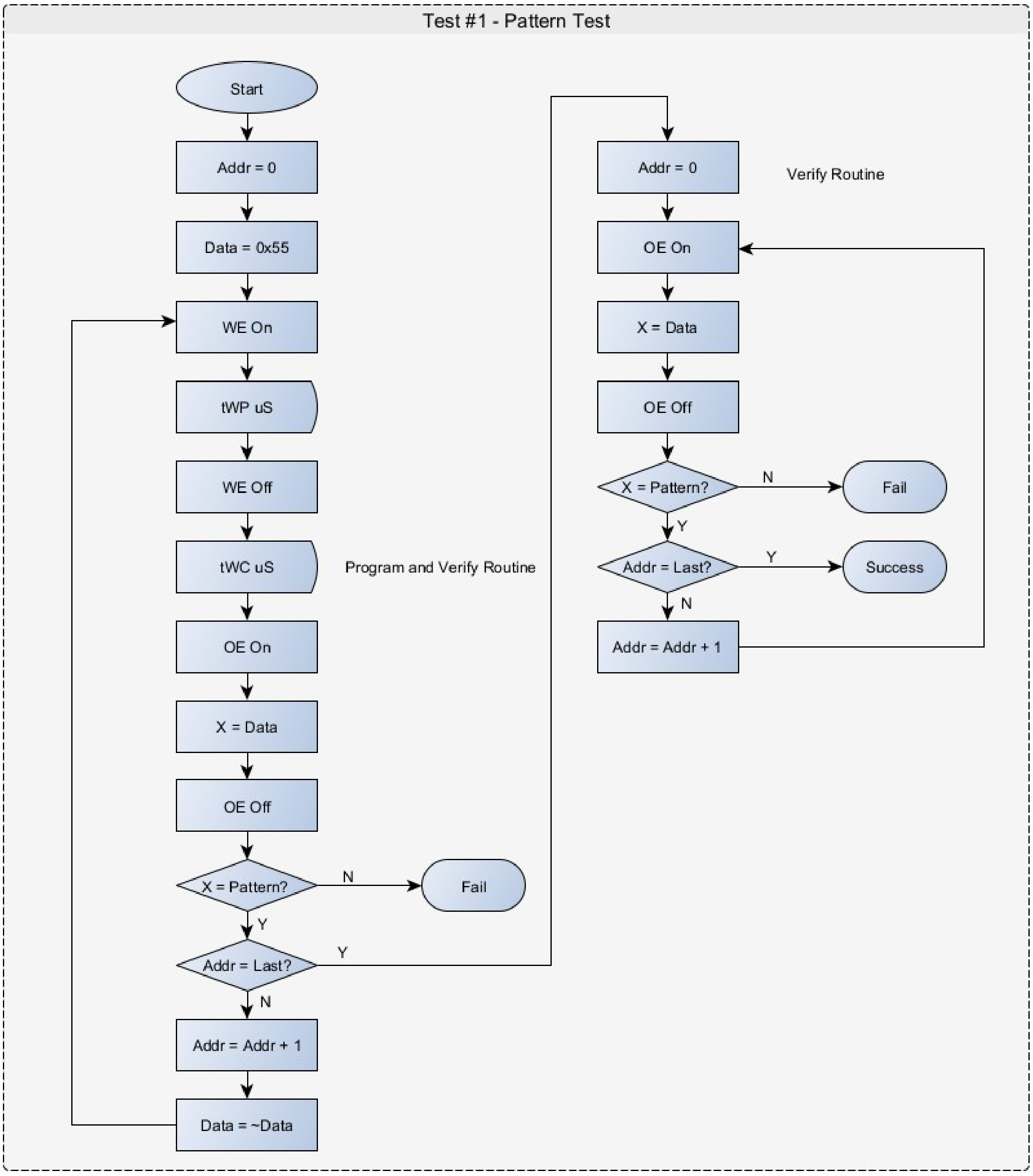


Figure 7: SRAM Pattern Test Routine

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

Test #2 Routine

The Test #2 routine is illustrated below.

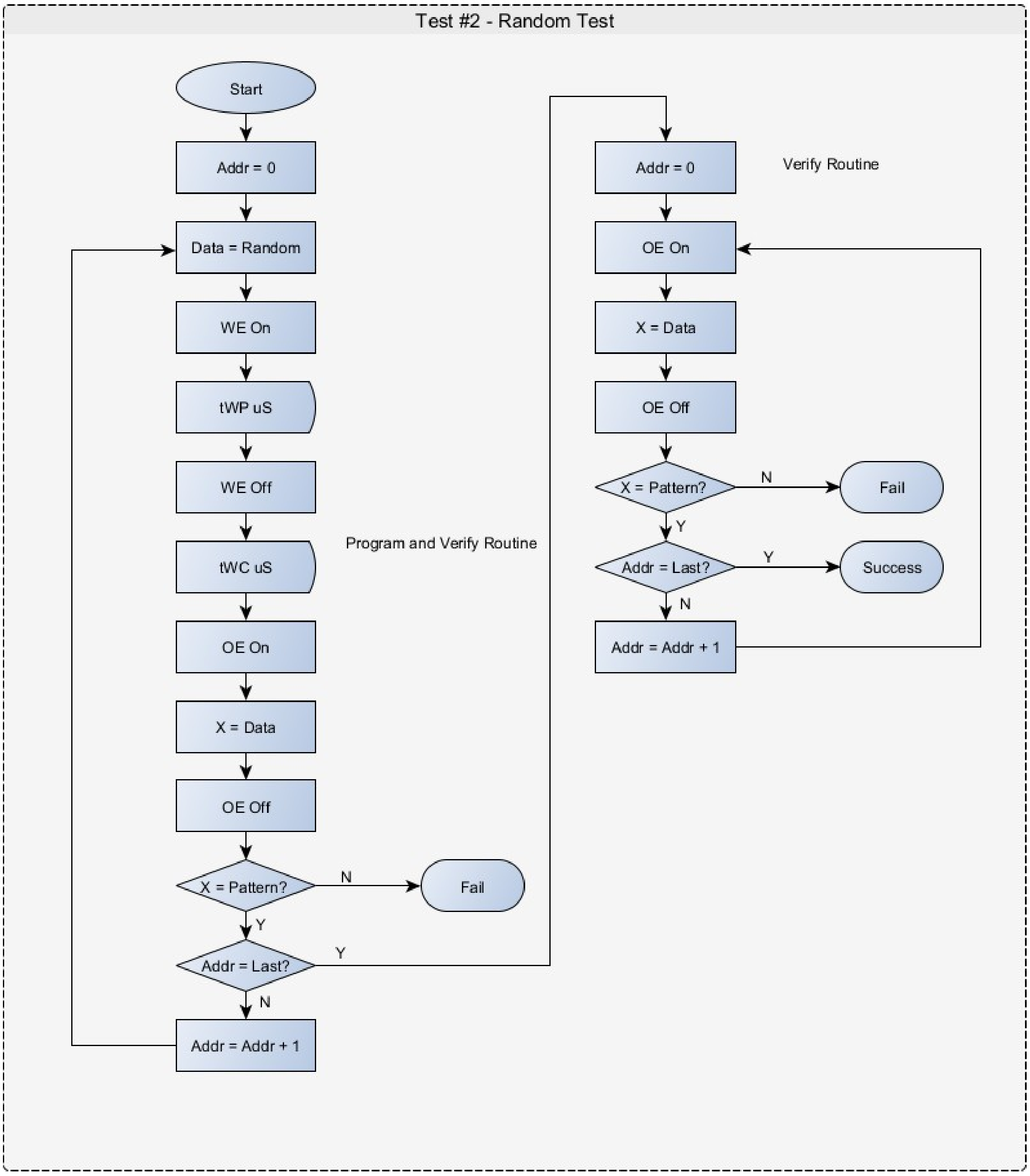


Figure 8: SRAM Random Test Routine

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

5.1.2. Parallel Memory – EPROM

An EPROM, or Erasable Programmable Read-Only Memory, is a type of programmable read-only memory (PROM) chip that retains its data when its power supply is switched off. Computer memory that can retrieve stored data after a power supply has been turned off and back on is called non-volatile. It’s an array of floating-gate transistors individually programmed by an electronic device that supplies higher voltages than those normally used in digital circuits. Once programmed, an EPROM can be erased by exposing it to strong ultraviolet (UV) light source (such as from a mercury-vapor lamp).

The most common EPROM chips operate on 5V power supply (VDD) to read, but can use a voltage other than VDD for programming (such as 6V), in addition to a VPP voltage that can vary between 12V and 25V, depending on the model.

Common pinouts

Considering most EPROMs encapsulated with DIP packages, the most common pinouts are shown below:

Table 9: EPROM DIP24 – Pinout

2532 (4K)	27x32 (4K)	27x16 (2K)		Chip		27x16 (2K)	27x32 (4K)	2532 (4K)
A7	A7	A7	1		24	VDD	VDD	VDD
A6	A6	A6	2		23	A8	A8	A8
A5	A5	A5	3		22	A9	A9	A9
A4	A4	A4	4		21	VPP	A11	$\overline{OE}$/VPP
A3	A3	A3	5		20	$\overline{OE}$	$\overline{OE}$/VPP	$\overline{CE}$/$\overline{PGM}$
A2	A2	A2	6		19	A10	A10	A10
A1	A1	A1	7		18	$\overline{CE}$/PGM	$\overline{CE}$/$\overline{PGM}$	A11
A0	A0	A0	8		17	D7	D7	D7
D0	D0	D0	9		16	D6	D6	D6
D1	D1	D1	10		15	D5	D5	D5
D2	D2	D2	11		14	D4	D4	D4
GND	GND	GND	12		13	D3	D3	D3

Table 10: EPROM DIP28 – Pinout

27x512 (64K)	27x256 (32K)	27x128 (16K)	27x64 (8K)		Chip		27x64 (8K)	27x128 (16K)	27x256 (32K)	27x512 (64K)
A15	VPP	VPP	VPP	1		28	VDD	VDD	VDD	VDD
A12	A12	A12	A12	2		27	$\overline{\mathrm{PGM}}$	$\overline{\mathrm{PGM}}$	A14	A14
A7	A7	A7	A7	3		26	NC	A13	A13	A13
A6	A6	A6	A6	4		25	A8	A8	A8	A8
A5	A5	A5	A5	5		24	A9	A9	A9	A9
A4	A4	A4	A4	6		23	A11	A11	A11	A11
A3	A3	A3	A3	7		22	$\overline{\mathrm{OE}}$	$\overline{\mathrm{OE}}$	$\overline{\mathrm{OE}}$	$\overline{\mathrm{OE}}$/VPP
A2	A2	A2	A2	8		21	A10	A10	A10	A10
A1	A1	A1	A1	9		20	$\overline{\mathrm{CE}}$	$\overline{\mathrm{CE}}$	$\overline{\mathrm{CE}/\mathrm{PGM}}$	$\overline{\mathrm{CE}/\mathrm{PGM}}$
A0	A0	A0	A0	10		19	D7	D7	D7	D7
D0	D0	D0	D0	11		18	D6	D6	D6	D6
D1	D1	D1	D1	12		17	D5	D5	D5	D5
D2	D2	D2	D2	13		16	D4	D4	D4	D4
GND	GND	GND	GND	14		15	D3	D3	D3	D3

Table 11: EPROM DIP32 – Pinout

27x080 (1M)	27x040 (512K)	27x020 (256K)	27x010 (128K)		Chip		27x010 (128K)	27x020 (256K)	27x040 (512K)	27x080 (1M)
A19	VPP	VPP	VPP	1		32	VDD	VDD	VDD	VDD
A16	A16	A16	A16	2		31	$\overline{\mathrm{PGM}}$	$\overline{\mathrm{PGM}}$	A18	A18
A15	A15	A15	A15	3		30	NC	A17	A17	A17
A12	A12	A12	A12	4		29	A14	A14	A14	A14
A7	A7	A7	A7	5		28	A13	A13	A13	A13
A6	A6	A6	A6	6		27	A8	A8	A8	A8
A5	A5	A5	A5	7		26	A9	A9	A9	A9
A4	A4	A4	A4	8		25	A11	A11	A11	A11
A3	A3	A3	A3	9		24	$\overline{\mathrm{OE}}$	$\overline{\mathrm{OE}}$	$\overline{\mathrm{OE}}$	$\overline{\mathrm{OE}}$/VPP
A2	A2	A2	A2	10		23	A10	A10	A10	A10
A1	A1	A1	A1	11		22	$\overline{\mathrm{CE}}$	$\overline{\mathrm{CE}}$	$\overline{\mathrm{CE}/\mathrm{PGM}}$	$\overline{\mathrm{CE}/\mathrm{PGM}}$
A0	A0	A0	A0	12		21	D7	D7	D7	D7
D0	D0	D0	D0	13		20	D6	D6	D6	D6
D1	D1	D1	D1	14		19	D5	D5	D5	D5
D2	D2	D2	D2	15		18	D4	D4	D4	D4
GND	GND	GND	GND	16		17	D3	D3	D3	D3

Table 12: EPROM (16Bit) DIP40 – Pinout

27x4096 (512K)	27x2048 (256K)	27x1024 (128K)		Chip		27x1024 (128K)	27x2048 (256K)	27x4096 (512K)
VPP	VPP	VPP	1		40	VDD	VDD	VDD
$\overline{\text{CE}}$/$\overline{\text{PGM}}$	$\overline{\text{CE}}$	$\overline{\text{CE}}$	2		39	$\overline{\text{PGM}}$	$\overline{\text{PGM}}$	A17
D15	D15	D15	3		38	NC	A16	A16
D14	D14	D14	4		37	A15	A15	A15
D13	D13	D13	5		36	A14	A14	A14
D12	D12	D12	6		35	A13	A13	A13
D11	D11	D11	7		34	A12	A12	A12
D10	D10	D10	8		33	A11	A11	A11
D9	D9	D9	9		32	A10	A10	A10
D8	D8	D8	10		31	A9	A9	A9
GND	GND	GND	11		30	GND	GND	GND
D7	D7	D7	12		29	A8	A8	A8
D6	D6	D6	13		28	A7	A7	A7
D5	D5	D5	14		27	A6	A6	A6
D4	D4	D4	15		26	A5	A5	A5
D3	D3	D3	16		25	A4	A4	A4
D2	D2	D2	17		24	A3	A3	A3
D1	D1	D1	18		23	A2	A2	A2
D0	D0	D0	19		22	A1	A1	A1
$\overline{\text{OE}}$	$\overline{\text{OE}}$	$\overline{\text{OE}}$	20		21	A0	A0	A0

Table 13: EPROM (16Bit) DIP42 – Pinout

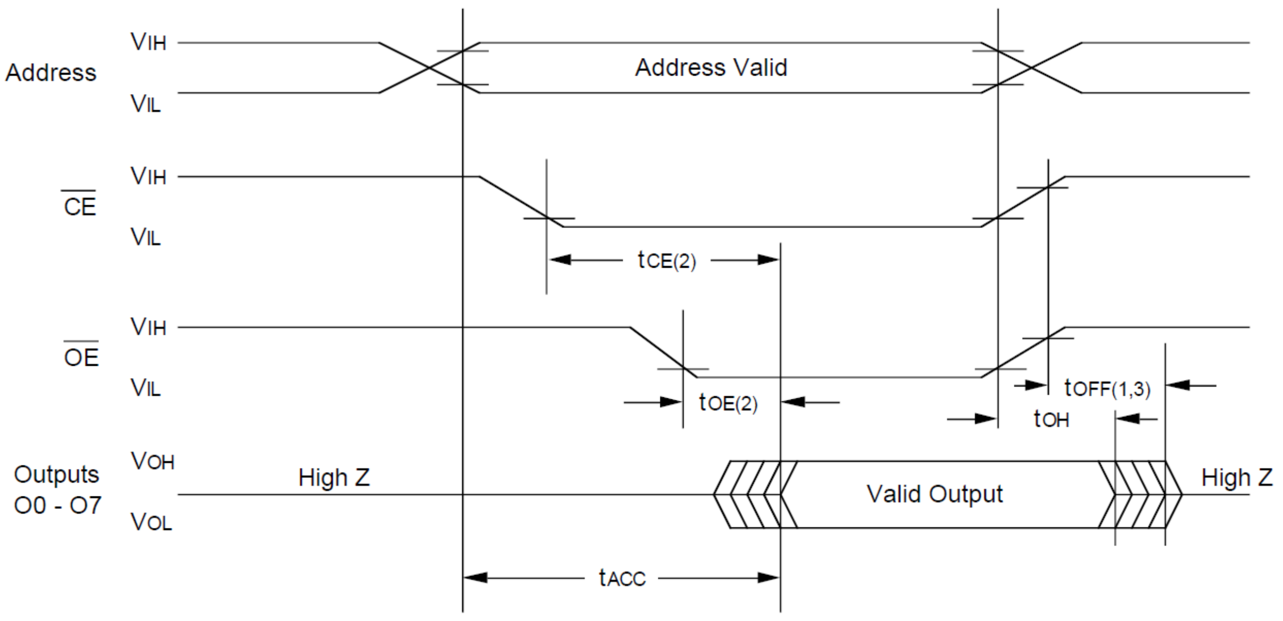
27x320 (4M)	27x160 (2M)	27x800 (1M)	27x400 (512K)			Chip		27x400 (512K)		27x800 (1M)	27x160 (2M)	27x320 (4M)
A18	A18	A18	DIP40		1		42	DIP40		NC	A19	A19
A17	A17	A17	A17	1	2		41	40	A8	A8	A8	A8
A7	A7	A7	A7	2	3		40	39	A9	A9	A9	A9
A6	A6	A6	A6	3	4		39	38	A10	A10	A10	A10
A5	A5	A5	A5	4	5		38	37	A11	A11	A11	A11
A4	A4	A4	A4	5	6		37	36	A12	A12	A12	A12
A3	A3	A3	A3	6	7		36	35	A13	A13	A13	A13
A2	A2	A2	A2	7	8		35	34	A14	A14	A14	A14
A1	A1	A1	A1	8	9		34	33	A15	A15	A15	A15
A0	A0	A0	A0	9	10		33	32	A16	A16	A16	A16
$\overline{\text{CE}}$/PGM	$\overline{\text{CE}}$/PGM	$\overline{\text{CE}}$/PGM	$\overline{\text{CE}/\text{PGM}}$	10	11		32	31	VPP	VPP	VPP	A20
GND	GND	GND	GND	11	12		31	30	GND	GND	GND	GND
$\overline{\text{OE}}$/VPP	$\overline{\text{OE}}$	$\overline{\text{OE}}$	$\overline{\text{OE}}$	12	13		30	29	D15	D15	D15	D15
D0	D0	D0	D0	13	14		29	28	D7	D7	D7	D7
D8	D8	D8	D8	14	15		28	27	D14	D14	D14	D14
D1	D1	D1	D1	15	16		27	26	D6	D6	D6	D6
D9	D9	D9	D9	16	17		26	25	D13	D13	D13	D13
D2	D2	D2	D2	17	18		25	24	D5	D5	D5	D5
D10	D10	D10	D10	18	19		24	23	D12	D12	D12	D12
D3	D3	D3	D3	19	20		23	22	D4	D4	D4	D4
D11	D11	D11	D11	20	21		22	21	VDD	VDD	VDD	VDD

Notes:

- The $\overline{\text{PGM}}$/$\overline{\text{CE}}$ pin is activated by the Programmer using the $\overline{\text{WE}}$ pin.
- The $\overline{\text{OE}}$/VPP pin is activated by the Programmer using the VPP pin.
- The $\overline{\text{OE}}$ pin is activated by the Programmer using the $\overline{\text{OE}}$ pin.
- The $\overline{\text{CE}}$ pin is activated by the Programmer using the $\overline{\text{CE}}$ pin.

Read Cycle

The EPROM read cycle can be illustrated:



Notes: (1) tOFF is specified for $\overline{OE}$ or $\overline{CE}$, whichever occurs first
(2) $\overline{OE}$ may be delayed up to tCE - tOE after the falling edge of $\overline{CE}$ without impact on tCE
(3) This parameter is sampled and is not 100% tested.

Figure 9: EPROM Read Cycle

To read an EPROM, the following steps are required:

1. Power the EPROM with VDD (for Read);
2. VPP pin (if any) must be with VDD;
3. $\overline{PGM}$ pin (if any) must be with HI (VDD) – for 27x16, $\overline{CE}$/PGM must be LO;
4. Put $\overline{OE}$ in HI (VDD);
5. Put $\overline{CE}$ in LO;
6. Put the address on bus A0..An;
7. Put $\overline{OE}$ in LO;
8. The data will be available on bus D0..Dn.
9. Put $\overline{OE}$ in HI (VDD);

Program Cycle

The EPROM program cycle can be illustrated:

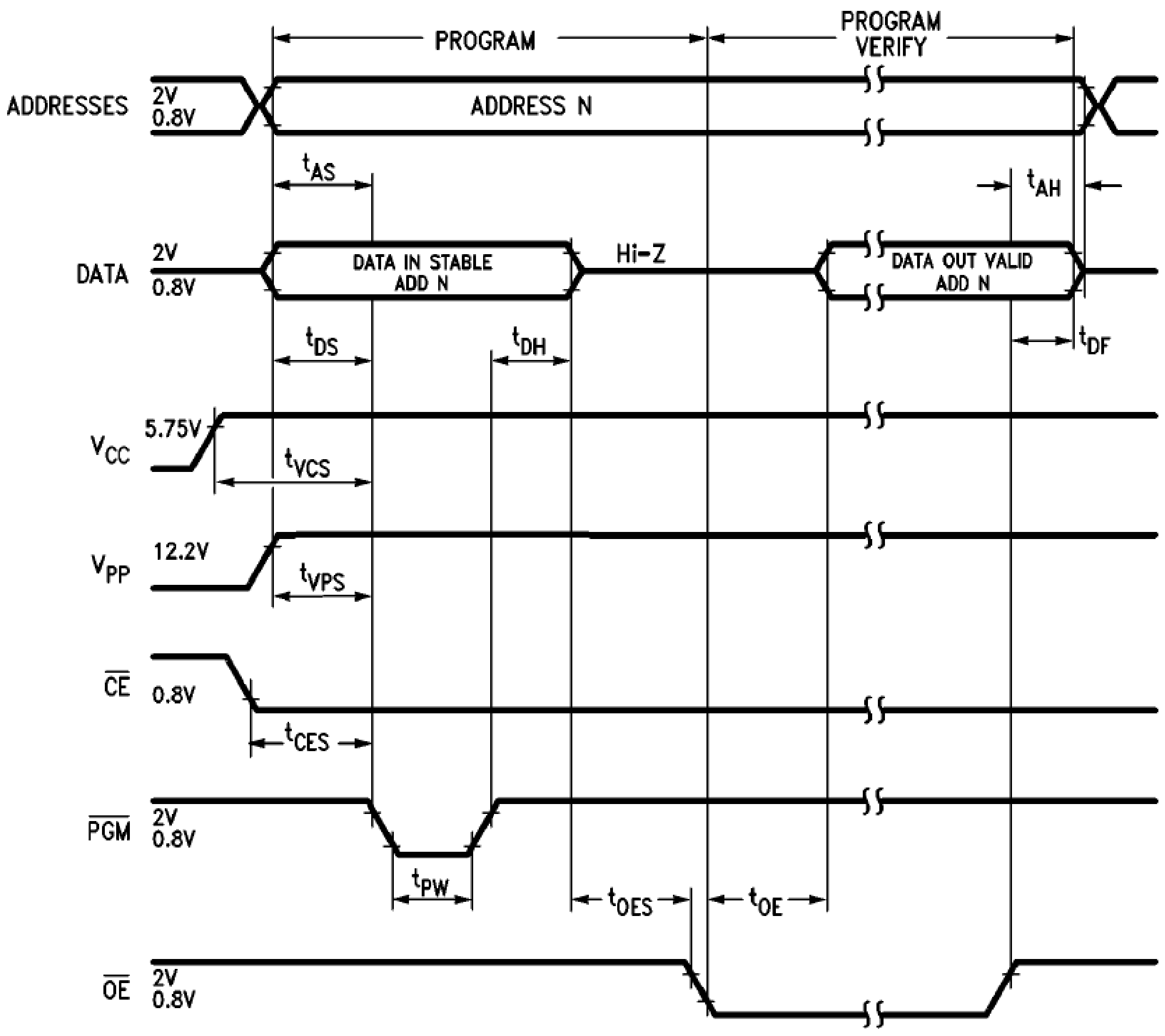


Figure 10: EPROM Program Cycle

To program an EPROM, the following steps are required:

1. Make sure the data is not 0xFF (only zero bits are written);
2. Power the EPROM with VDD (to Program);
3. Put $\overline{\text{PGM}}$, $\overline{\text{OE}}$, $\overline{\text{CE}}$ in HI (VDD) – for 27x16, $\overline{\text{CE}}$/PGM must be LO;
4. Put $\overline{\text{CE}}$ in LO;
5. Power the EPROM with VPP;
6. Put the address on bus A0..An;
7. Put the data on bus D0..Dn;

8. Put $\overline{\text{PGM}}$ in LO – for 27x16, $\overline{\text{CE}}$/PGM must be HI (VDD);
9. Wait for tWP time;
10. Put $\overline{\text{PGM}}$ in HI (VDD) – for 27x16, $\overline{\text{CE}}$/PGM must be LO;
11. The data will be recorded in memory;
12. VPP pin (if any) must be with VDD.

To verify data recorded in the same cycle:

1. VPP pin (if any) must be with VDD;
2. $\overline{\text{PGM}}$ pin (if any) must be with HI (VDD) – for 27x16, $\overline{\text{CE}}$/PGM must be LO;
3. Put $\overline{\text{OE}}$ in LO;
4. The data will be available on bus D0..Dn;
5. Put $\overline{\text{OE}}$ in HI (VDD).

EPROM Read Algorithm

The EPROM reading algorithm is proposed as follows.

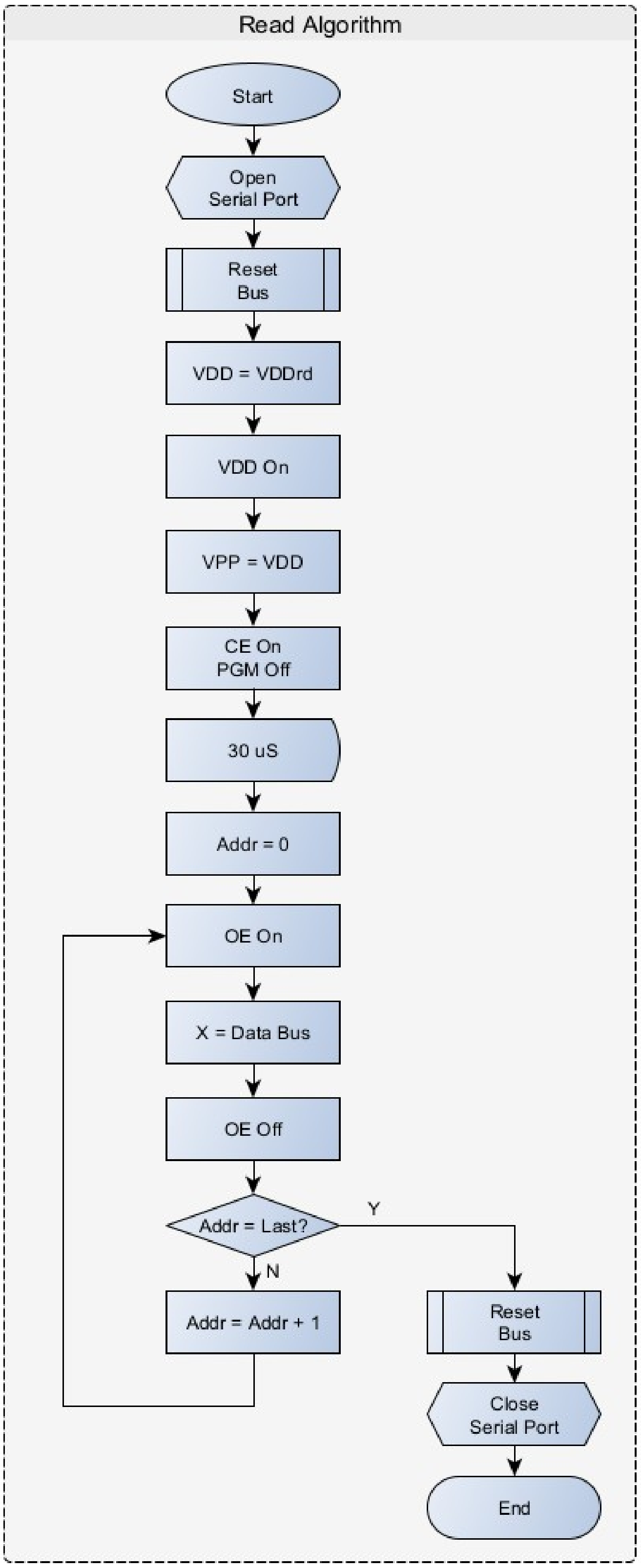


Figure 11: EPROM Read Algorithm

EPROM Program Algorithm

The EPROM programming algorithm is proposed as follows.

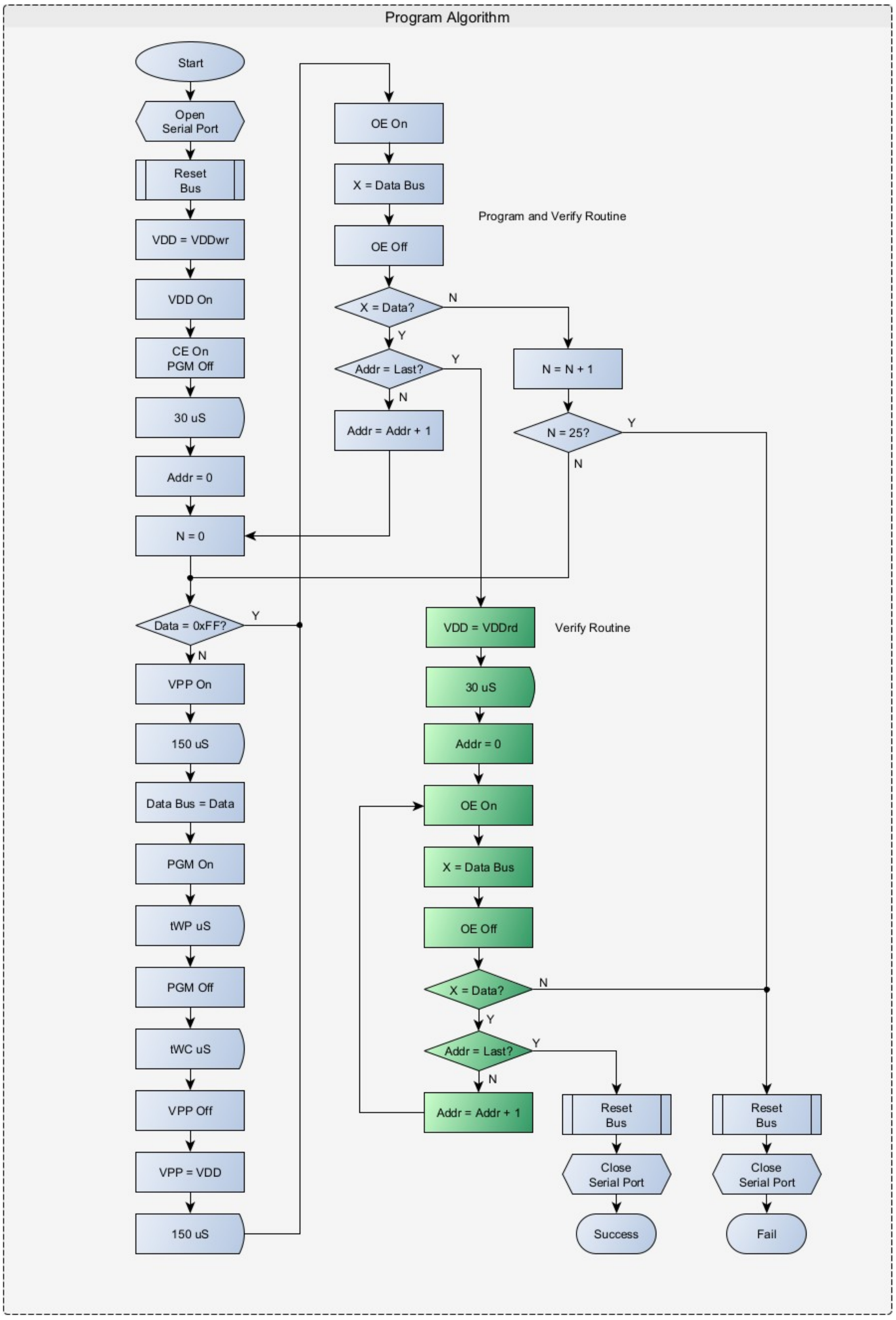


Figure 12: EPROM Program Algorithm

EPROM GetID Algorithm

The EPROM getting ID algorithm is proposed as follows.

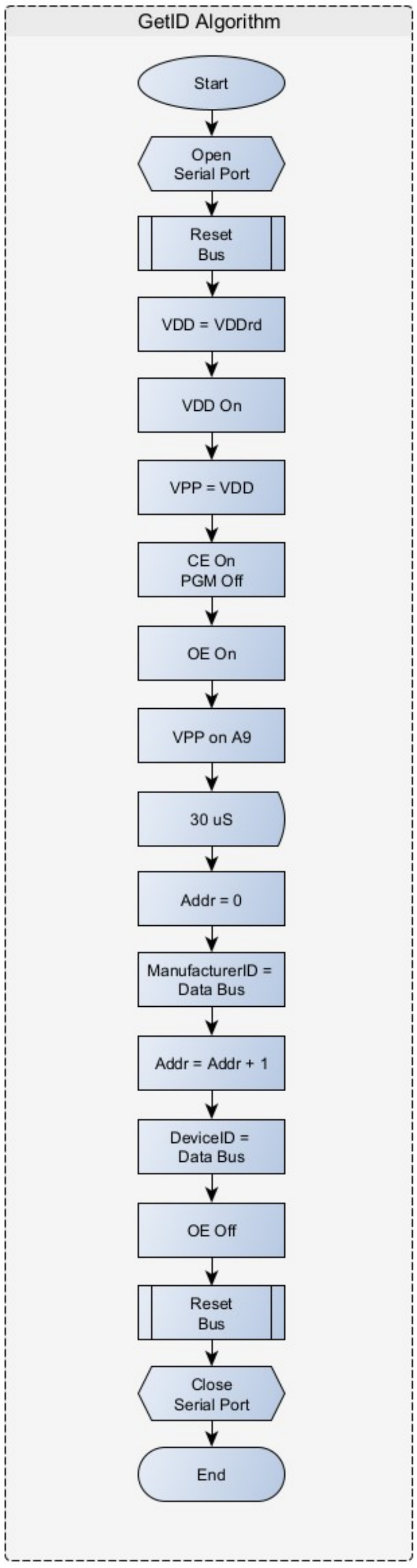


Figure 13: EPROM GetID Algorithm

Reset Bus Routine

The Reset Bus routine is illustrated below.

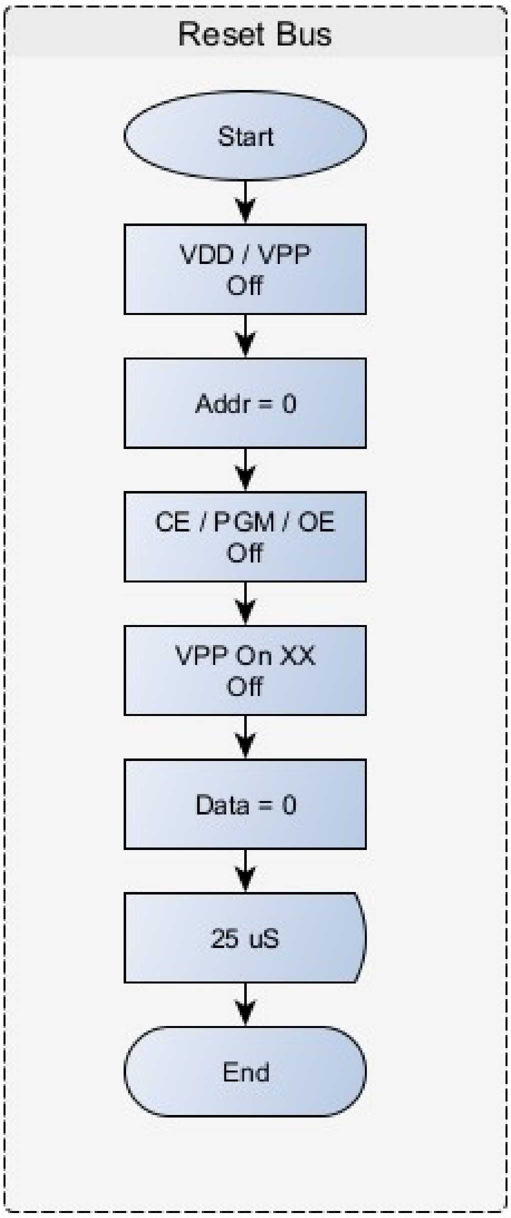


Figure 14: EPROM Reset Bus Routine

5.1.3. Parallel Memory – Electrically Erasable EPROM

An Electrically Erasable EPROM is similar to a conventional CMOS EPROM, but instead of being erased using ultraviolet (UV) light, it can be erased using an electrical pulse.

Common pinouts

The pinouts are the same as conventional EPROMs.

Read Cycle

The reading cycle is the same as conventional EPROMs.

Program Cycle

The programming cycle is the same as conventional EPROMs.

Erase Cycle

The EPROM erase cycle can be illustrated:

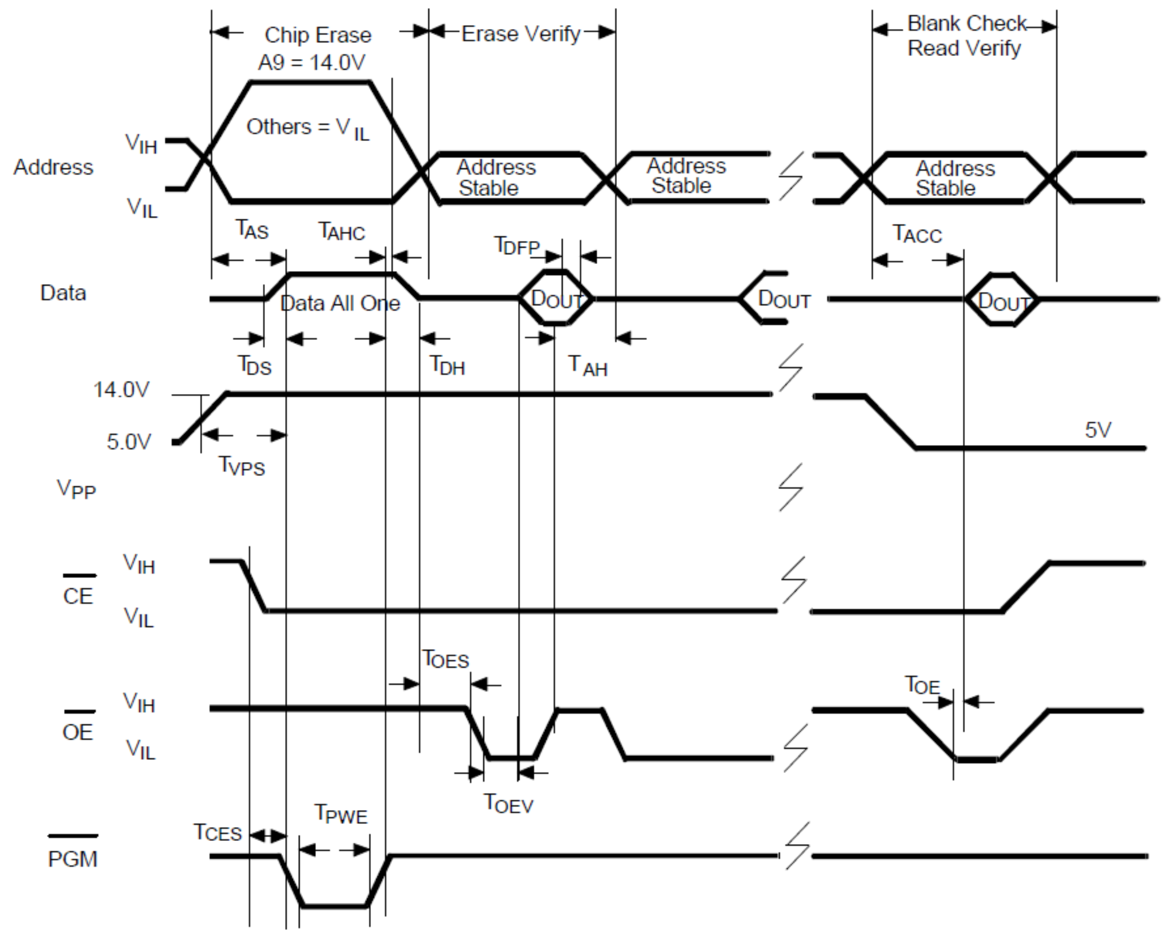


Figure 15: EPROM Erase Cycle

To erase an EPROM, the following steps are required:

1. Power the EPROM with VDD (to Program);
2. Put $\overline{\text{PGM}}$, $\overline{\text{OE}}$, $\overline{\text{CE}}$ in HI (VDD);
3. Put the VPP pin with VEE;
4. Put $\overline{\text{CE}}$ in LO;
5. Put the address bus with 0x00;
6. Put the A9 pin with VEE;
7. Put the data bus with 0xFF;
8. Put $\overline{\text{PGM}}$ in LO;
9. Wait for tPWE time (100 ms);

10. Put $\overline{\text{PGM}}$ in HI (VDD);
11. The all data will be erased.

To verify if data is erased in the same cycle:

1. VPP pin must be with VEE;
2. $\overline{\text{PGM}}$ pin must be with HI (VDD);
3. $\overline{\text{CE}}$ pin must be with LO;
4. Put the address in bus A0..An;
5. Put $\overline{\text{OE}}$ in LO;
6. The data will be available on bus D0..Dn;
7. Put $\overline{\text{OE}}$ in HI (VDD);
8. Check if data is 0xFF. If yes, read the next address (step 4). If no, repeat the erase cycle above (step 5). Do this for up to 20 attempts. If it fails, the device has a problem.

EPROM Read Algorithm

The reading algorithm is the same as conventional EPROMs.

EPROM Program Algorithm

The programming algorithm is the same as conventional EPROMs.

EPROM GetID Algorithm

The getting ID algorithm is the same as conventional EPROMs.

EPROM Erase Algorithm

The EPROM erasing algorithm is proposed as follows.

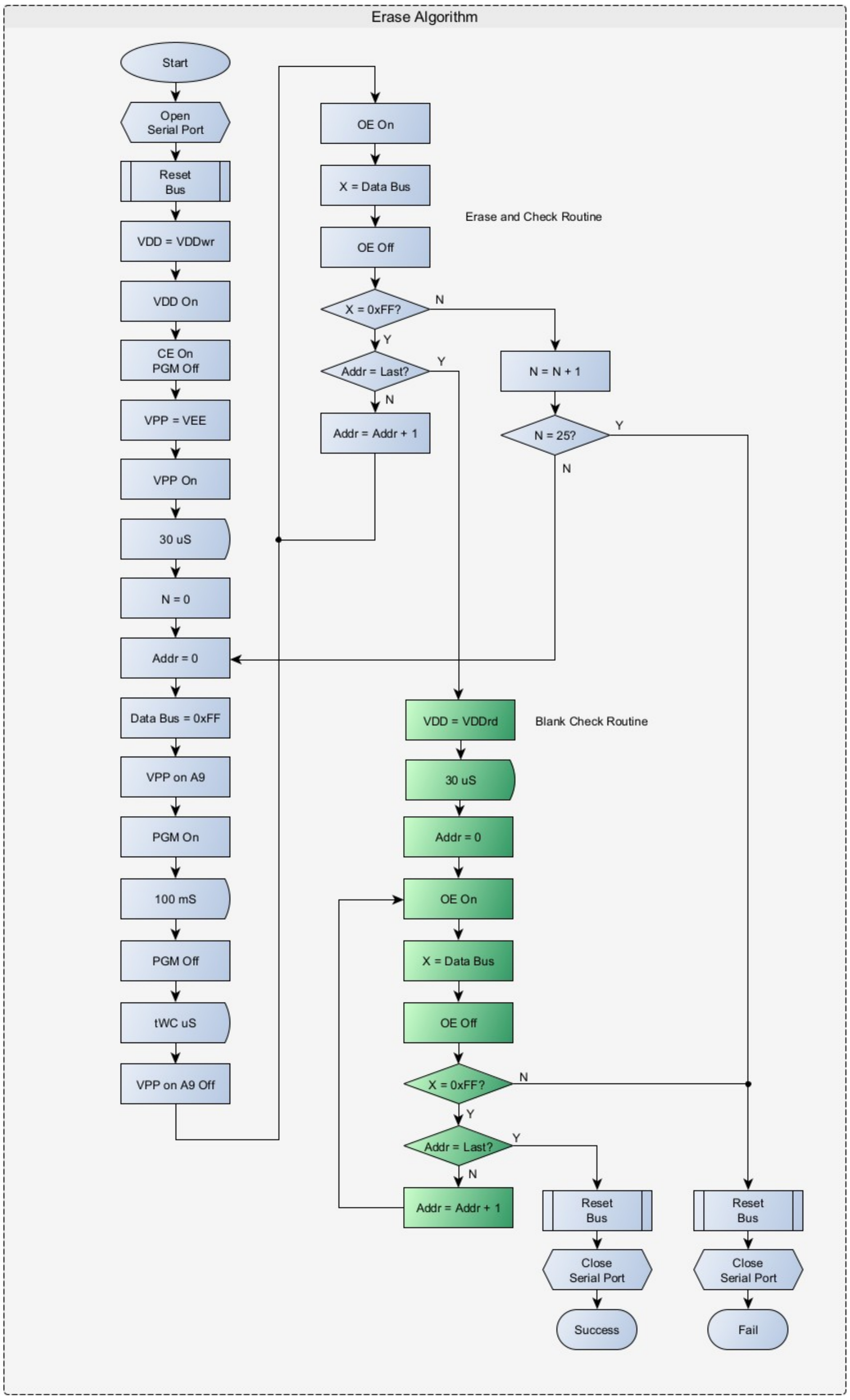


Figure 16: EPROM Erase Algorithm

5.1.4. Parallel Memory – EEPROM

An EEPROM or E^{2}PROM (Electrically Erasable Programmable Read-Only Memory) is a type of non-volatile memory. EEPROMs are organized as arrays of floating-gate transistors, and can be programmed and erased in-circuit, by applying special programming signals.

Originally, EEPROMs were limited to single-byte operations, which made them slower, but modern EEPROMs allow multi-byte page operations.

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

Common pinouts

The pinouts are the same as SRAMs.

Read Cycle

The reading cycle is the same as SRAMs.

Program Cycle

The EEPROM program cycle can be illustrated:

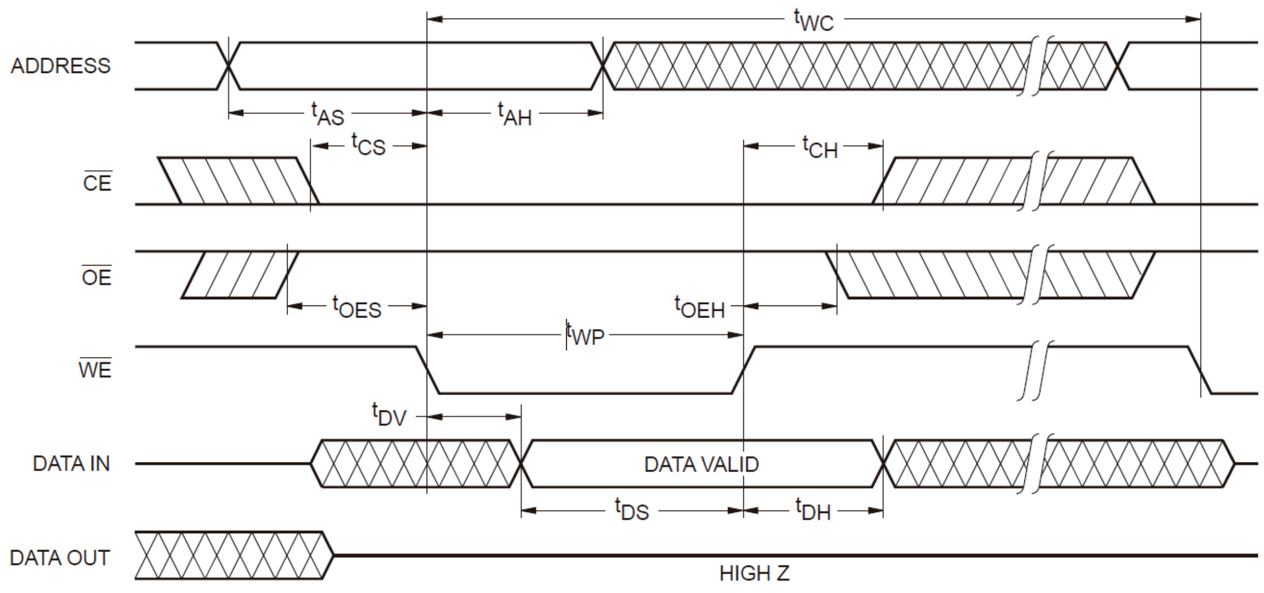


Figure 17: EEPROM Program Cycle (Byte Write)

To program an EEPROM (byte mode), the following steps are required:

1. Power the EEPROM with VDD;
2. Wait the stabilization time (~1 to 5 ms);
3. Put $\overline{WE}$, $\overline{OE}$, $\overline{CE}$ in HI;
4. Put $\overline{CE}$ in LO;
5. Put the address on bus A0..An;
6. Put the data on bus D0..Dn;
7. Put $\overline{WE}$ in LO;
8. Wait for tWP time;
9. Put $\overline{WE}$ in HI;
10. Wait for tWC time (or do Data Pooling);
11. The data will be recorded in memory.

To verify data recorded in the same cycle:

1. $\overline{WE}$ pin must be with HI;
2. Put $\overline{OE}$ in LO;
3. The data will be available on bus D0..Dn;
4. Put $\overline{OE}$ in HI.

Program Cycle (Page Write)

Some EEPROMs support page writing (such as the Atmel AT28C). This EEPROM page writing cycle can be illustrated:

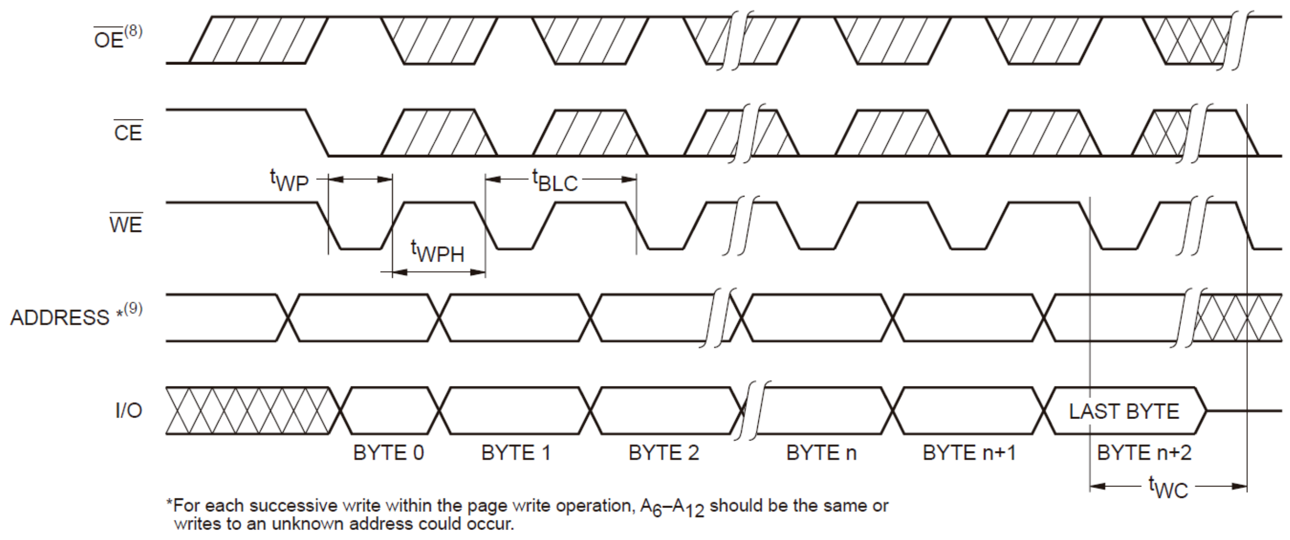

OE(8)
CE
tWP
tBLC
WE
tWPH
ADDRESS *(9)
I/O
LAST BYTE
BYTE 0
BYTE 1
BYTE 2
BYTE n
BYTE n+1
BYTE n+2
tWC
*For each successive write within the page write operation, A6–A12 should be the same or writes to an unknown address could occur.

Figure 18: EEPROM Page Write Cycle

For the device to operate in page write mode, more than one byte must be written to a sequential address (on the same page), within a short time interval (a few microseconds), instead of waiting for tWC before writing the next byte.

Some devices support between 2 to 16 bytes per page, while others support higher values, such as 64, 128 up to 2KB.

Erase Cycle

Most EEPROMs do not have a special erase cycle. To erase all content, simply write 0xFF to all memory addresses.

EEPROM Read Algorithm

The EEPROM reading algorithm is proposed as follows.

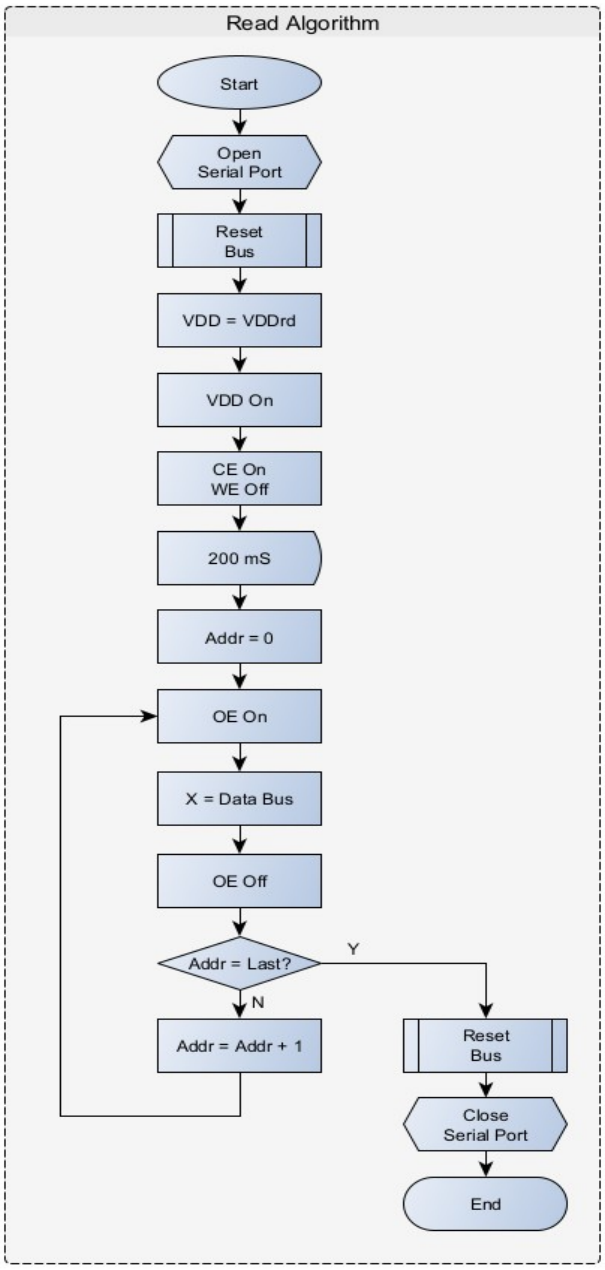


Figure 19: EEPROM Read Algorithm

EEPROM Program Algorithm (Byte)

The EEPROM programming algorithm is proposed as follows.

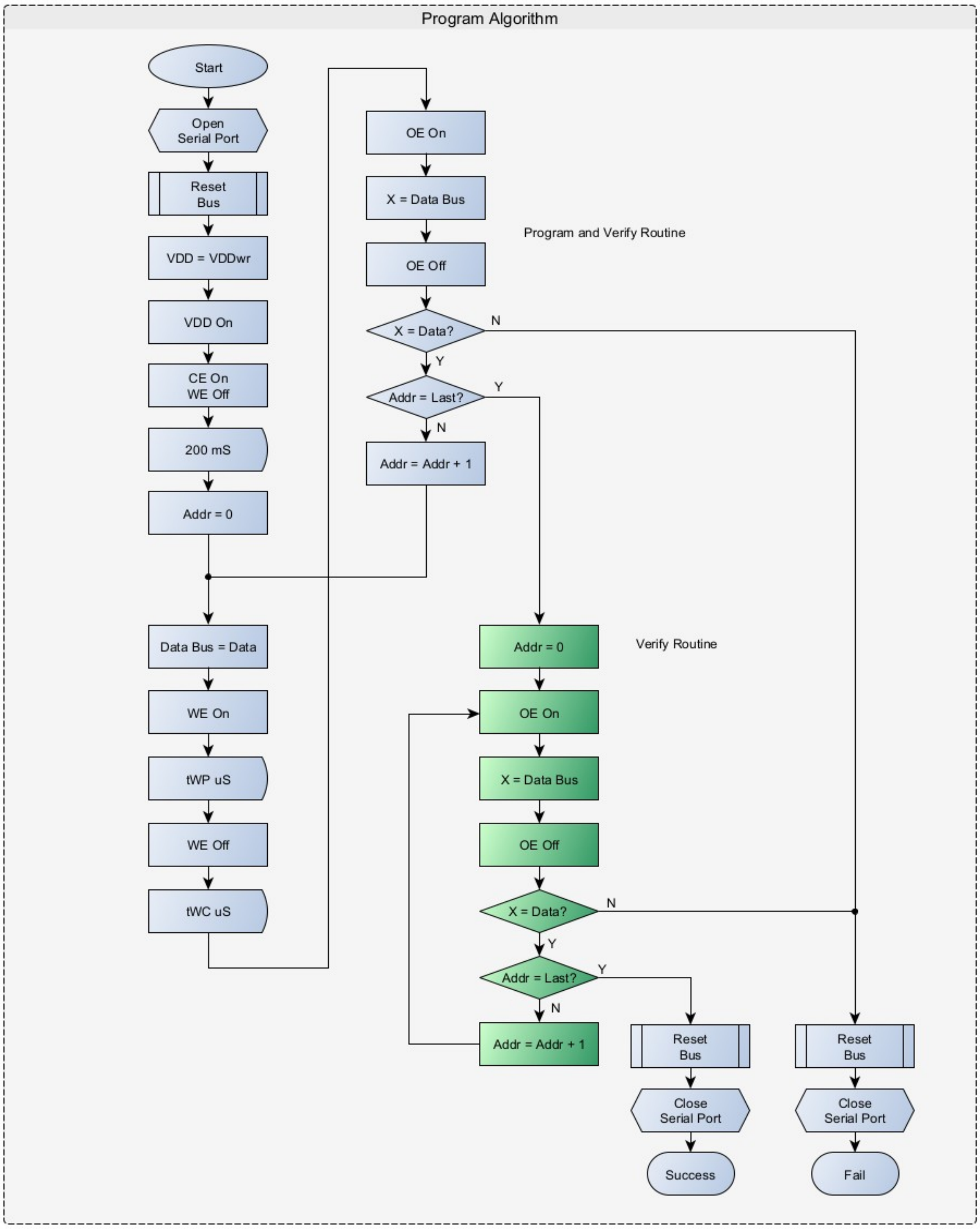


Figure 20: EEPROM Program Algorithm

To program in Page Write mode, simply write all bytes that are part of the page in sequence, without waiting for the tWC time (waiting for something short like tWP). At the end of the page, wait for the tWC time to complete the programming.

EEPROM Erase Algorithm

It's the same programming algorithm, writing 0xFF to all memory addresses.

EEPROM Software Unprotect Algorithm

The EEPROM software unprotect algorithm is proposed as follows.

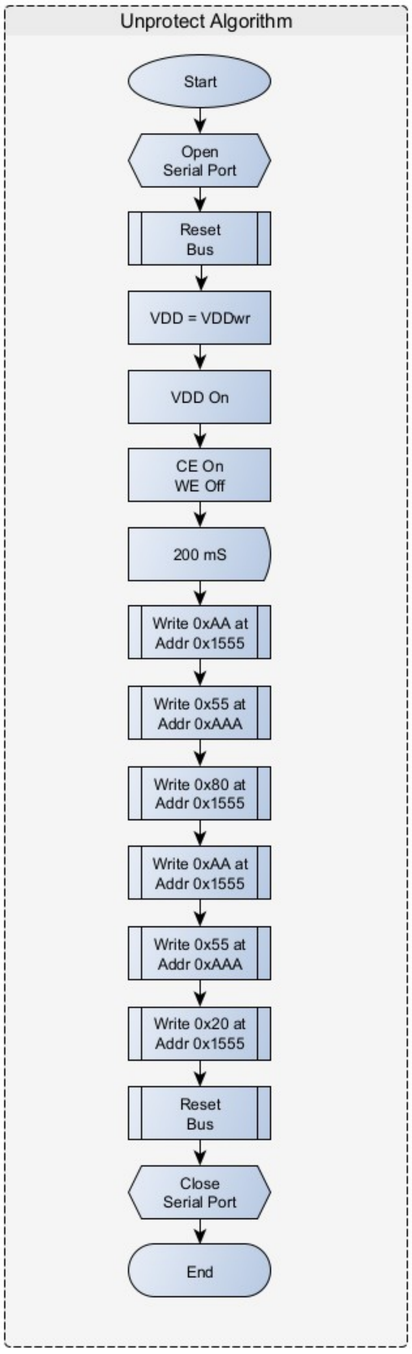


Figure 21: EEPROM Unprotect Algorithm

Note: Not all devices support this.

EEPROM Software Protect Algorithm

The EEPROM software protect algorithm is proposed as follows.

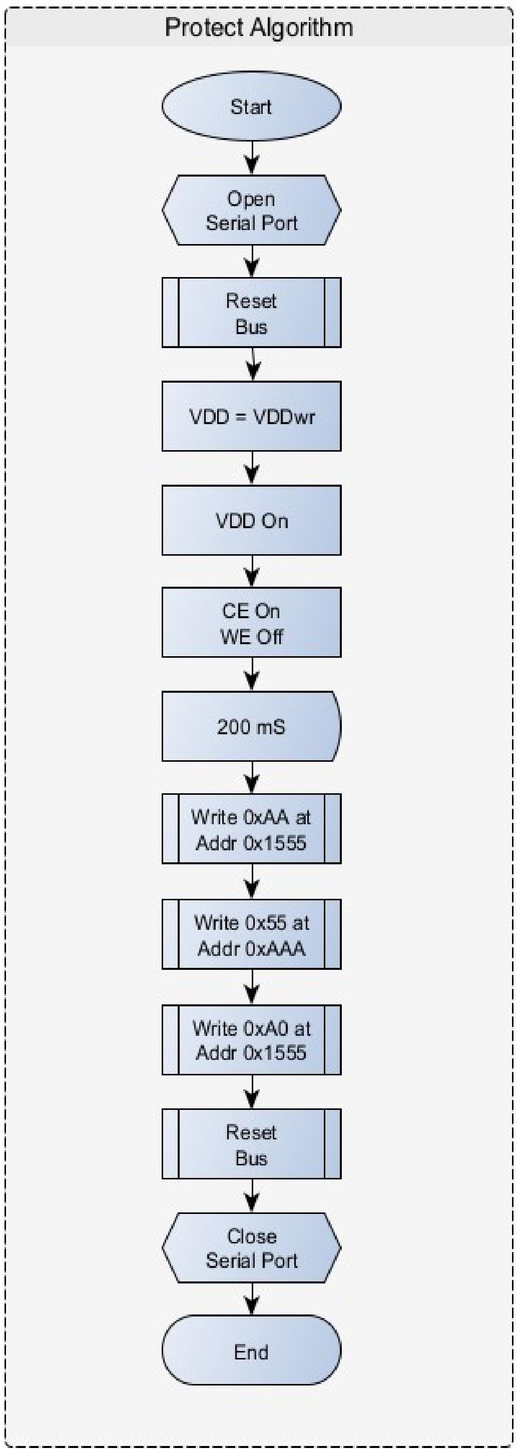


Figure 22: EEPROM Protect Algorithm

Note: Not all devices support this.

Reset Bus Routine

The Reset Bus routine is illustrated below.

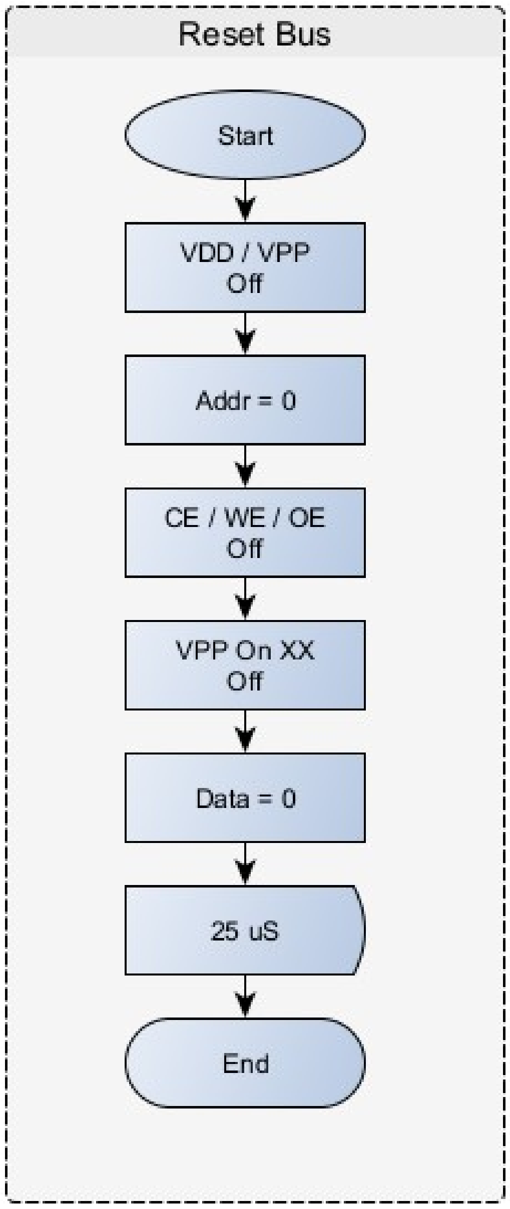


Figure 23: EEPROM Reset Bus Routine

Appendix A – Pinout Compatibility with EzoFlash+

This section shows the pinout compatibility of the adapter connectors between the USB Flash/EPROM Programmer and the EzoFlash+ (v.4.4/v.4.5):

A.1. USB Flash/EPROM Programmer Parallel Adapter Connector

Female (Top Side)

1	3	5	7	9	11	13	15	17	19	21[1]	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[1]	36	38	40	42	44	46	48

EzoFlash+ BU3 Female (Top Side)

1	3	5	7	9	11	13	15	17	19	21[1]	23	25	27	29	31
2	4	6	8	10	12	14	16	18	20	22	24	26	28	30	32

EzoFlash+ BU4 Female (Top Side)

33
34
35
36
37

Table 14: Parallel Adapter Connector x EzoFlash+

Pin	Function	Description	EzoFlash+
1	D0	DATA BUS – BIT 0	BU3 – 1
2	D1	DATA BUS – BIT 1	BU3 – 2
3	D2	DATA BUS – BIT 2	BU3 – 3
4	GND	GROUND	BU3 – 4
5	D3	DATA BUS – BIT 3	BU3 – 5
6	D4	DATA BUS – BIT 4	BU3 – 6
7	D5	DATA BUS – BIT 5	BU3 – 7
8	D6	DATA BUS – BIT 6	BU3 – 8
9	D7	DATA BUS – BIT 7	BU3 – 9
10	$\overline{\text{CE}}$	$\overline{\text{CHIP ENABLE}}$	BU3 – 10
11	A10	ADDRESS BUS – BIT 10	BU3 – 11
12	$\overline{\text{OE}}$	$\overline{\text{OUTPUT ENABLE}}$	BU3 – 12
13	A11	ADDRESS BUS – BIT 11	BU3 – 13
14	A9	ADDRESS BUS – BIT 9	BU3 – 14
15	A8	ADDRESS BUS – BIT 8	BU3 – 15

Pin	Function	Description	EzoFlash+
16	A13	ADDRESS BUS – BIT 13	BU3 – 16
17	A14	ADDRESS BUS – BIT 14	BU3 – 17
18	A17	ADDRESS BUS – BIT 17	BU3 – 18
19	$\overline{\text{WE}}$	$\overline{\text{WRITE ENABLE}}$	BU3 – 19
20	VDD	VDD VOLTAGE	BU3 – 20
21	A18	ADDRESS BUS – BIT 18	BU3 – 21 (JP5) [1]
22	A16	ADDRESS BUS – BIT 16	BU3 – 22
23	A15	ADDRESS BUS – BIT 15	BU3 – 23
24	A12	ADDRESS BUS – BIT 12	BU3 – 24
25	A7	ADDRESS BUS – BIT 7	BU3 – 25
26	A6	ADDRESS BUS – BIT 6	BU3 – 26
27	A5	ADDRESS BUS – BIT 5	BU3 – 27
28	A4	ADDRESS BUS – BIT 4	BU3 – 28
29	A3	ADDRESS BUS – BIT 3	BU3 – 29
30	A2	ADDRESS BUS – BIT 2	BU3 – 30
31	A1	ADDRESS BUS – BIT 1	BU3 – 31
32	A0	ADDRESS BUS – BIT 0	BU3 – 32
33	*KEY*	*KEY TO AVOID CONNECTOR INVERSION*	-
34	VPP	VPP PROGRAMMING VOLTAGE	BU4 – 37
			BU3 – 21 (JP4) [1]
35	A19	ADDRESS BUS – BIT 19	BU4 – 33
36	A20	ADDRESS BUS – BIT 20	BU4 – 34
37	A21	ADDRESS BUS – BIT 21	BU4 – 35
38	A22	ADDRESS BUS – BIT 22	BU4 – 36
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	-

1 Note that pin 21 of BU3 must be connected to the center of jumpers JP4 and JP5. If JP4 is placed, you must connect pin BU3-21 to the VPP signal. If JP5 is placed, you must connect pin BU3-21 to A18 signal.

A.2. USB Flash/EPROM Programmer Serial Adapter Connector

Female (Top Side)

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

EzoFlash+ (v.4.5) ICSP-SER Male (Top Side)

8	7[2]	6[2]	5	4[2]	3	2	1

Table 15: Serial Adapter Connector x EzoFlash+ v.4.5 Pin-Out

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

2 Note that pins 11 and 12 of the connector (pins 4/6 and 7 of ICSP-SER) must be joined in the case of memories that have a single data line (SDA), such as 24Cxx (I2C). To do this, it's advisable that there is a jumper (JP I2C), which, if placed, joins these 2 pins.

Appendix B – Pinout Compatibility with PK2C/MPSP

This section shows the pinout compatibility of the adapter connectors between the USB Flash/EPROM Programmer and the PK2C (v.1.0) and MPSP (v.1.1):

B.1. USB Flash/EPROM Programmer 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

PK2C (v.1.0) / MPSP (v.1.1) Female (Top Side)

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

Table 16: Serial Adapter Connector x PK2C/MPSP Pin-Out

Pin	Function	Description	PK2C/MPSP
1	VPP13	VPP PROGRAMMING VOLTAGE (13V)	J3 – 1
2	VPP13	VPP PROGRAMMING VOLTAGE (13V)	J3 – 2
3	VDD5	VDD VOLTAGE (5V)	J3 – 3
4	VDD5	VDD VOLTAGE (5V)	J3 – 4
5	GND	GROUND	J3 – 5
6	GND	GROUND	J3 – 6
7	SCK	SERIAL CLOCK	J3 – 7
8	SCK	SERIAL CLOCK	J3 – 8
9	GND	GROUND	J3 – 9
10	GND	GROUND	J3 – 10
11	SDO	SERIAL DATA OUT (TO TARGET SDI)	J3 – 11
12	SDI	SERIAL DATA IN (FROM TARGET SDO)	J3 – 12
13	GND	GROUND	J3 – 13
14	GND	GROUND	J3 – 14
15	VCC	VCC SUPPLY (5V – ALWAYS ON)	J3 – 15
16	VCC	VCC SUPPLY (5V – ALWAYS ON)	J3 – 16
17	$\overline{WE}$	CHIP SELECT ($\overline{WE}$ PROGRAMMER PIN)	J3 – 17
18	$\overline{WE}$	CHIP SELECT ($\overline{WE}$ PROGRAMMER PIN)	J3 – 18
19	GND	GROUND	J3 – 19
20	*KEY*	*KEY TO AVOID CONNECTOR INVERSION*	J3 – 20

Appendix C – 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/)