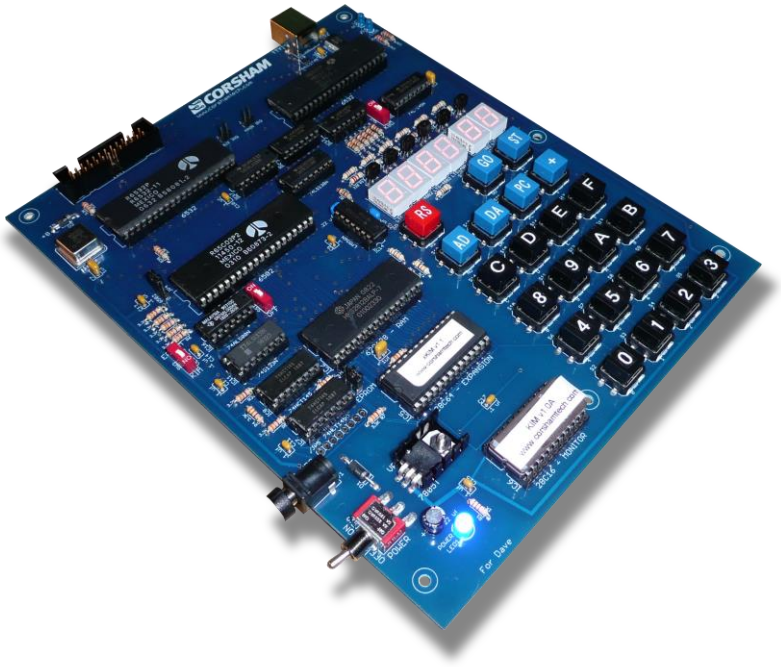
A PROGRAMMER'S GUIDE TO KIM PROGRAMMING

1 - DIE HARDWARE ORGANISATION DES 6502



Wie schaut es im Inneren aus

Wie ich es schon angedroht hatte, ist es doch notwendig, sich ein bisschen über die innere Struktur eines 6502 klar zu sein. Natürlich werde ich nicht bis ins kleinste Detail alle Elemente dieser CPU benennen, denn es würde den Rahmen sprengen und ist zudem auch nicht unbedingt erforderlich. Schauen wir uns also die Systemarchitektur genauer an.

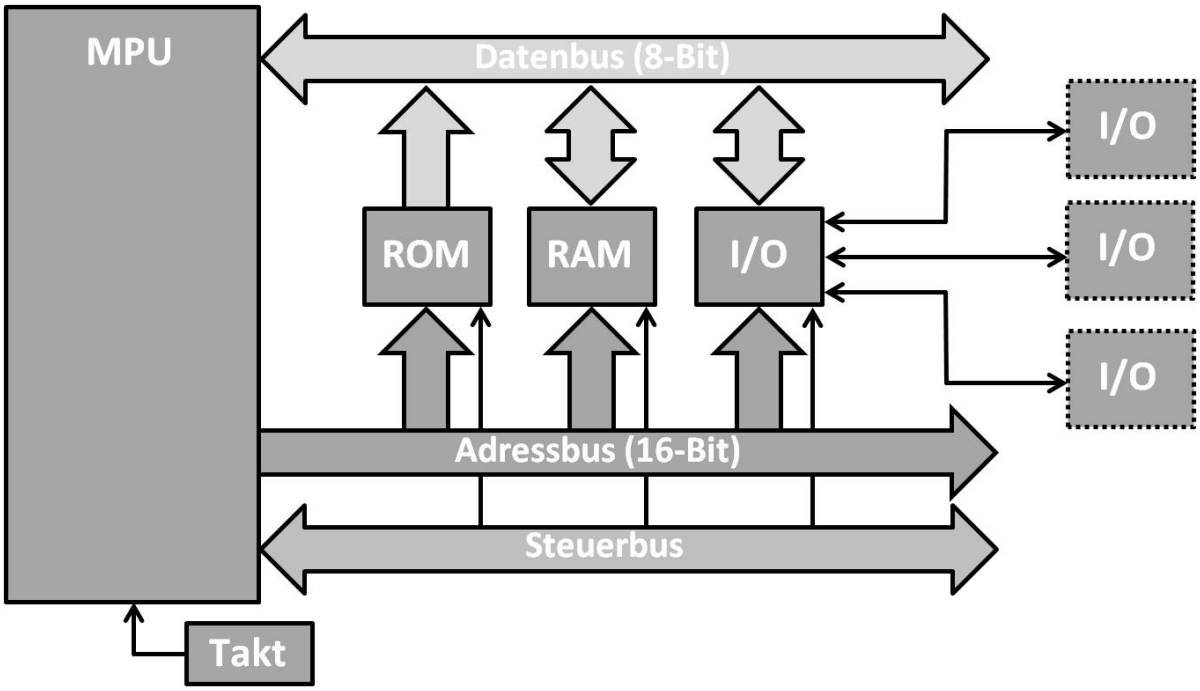


Abbildung 1: Die Systemarchitektur des 6502

Wenn es lautet, dass die 6502 CPU ein 8-Bit Mikroprozessor ist, dann bezieht sich das auf die Breite des Datenbusses. In der Abbildung sind drei verschiedene Bussysteme zu sehen und sie haben die Aufgabe, dass die verschiedenen Systemeinheiten wie Festwertspeicher (ROM - Read Only Memory), Arbeitsspeicher (RAM - Random Access Memory) und Ein- bzw. Ausgabeeinheit (I/O) miteinander kommunizieren bzw. Daten austauschen können. Was ist die Aufgabe im Einzelnen?

Der Datenbus

Zwischen den einzelnen Systemeinheiten müssen natürlich Daten hin- und hergeschoben werden, also ein reger Austausch stattfinden. Für diese Aufgabe ist der Datenbus mit einer Datenbreite von 8 Bit zuständig. Dieser Bus kann sowohl von der MPU (Micro Processing Unit) zu den Systemkomponenten die Daten verschicken, als auch Daten von ihnen empfangen. Der Datenfluss geht also in beiden Richtungen von statten, was bi-direktional genannt wird. Wieviel Zustände können eigentlich mit 8 Bits erzeugt werden? Das lässt sich über die folgende kurze Rechnung ermitteln:

$$2^8 = 256$$

Mit 8 Datenleitungen sind also 256 unterschiedliche Zustände möglich. Die 8 Bits werden in der Computertechnik zu einer größeren Einheit zusammengefasst, was 1 Byte genannt wird.

Der Adressbus

Damit die Daten ihren richtigen Weg über die verschiedenen Systemeinheiten finden - also von wo nach wo - muss die richtige Adresse angesprochen werden. Sie ist vergleichbar mit einer Hausnummer einer Straße, die im Normalfall nur ein einziges Mal vergeben ist. Um also eine bestimmte Speicherstelle in den Systemeinheiten anzusprechen, wird von der MPU über ein bestimmtes Register eine Adresse generiert und diese auf den Adressbus gelegt. Die Richtung der Adressierung erfolgt immer von der MPU zu den Systemeinheiten und ist quasi eine Einbahnstraße. Aus diesem Grund wird dies uni-direktional genannt. Wieviel Zustände können denn nun mit 16 Bits erzeugt werden?

$$2^{16} = 65.536$$

Mit 16 Adressleitungen sind also 65.536 unterschiedliche Zustände möglich. Wie viel Bytes sind das aber? Nun, wenn man auf die nächst höhere Einheit geht, also *Kilo*, dann sind das normalerweise 1.000. Hier läuft das jedoch ein wenig anders ab, denn 1 Kilo-Byte entsprechen nicht 1.000 Bytes! Die folgende Rechnung zeigt es uns:

$$2^{10} = 1.024$$

Warum ist das so? Wir haben es hier mit einem Binärsystem zu tun und die Basis ist eben die 2 und nicht wie bei unserem 10er-System die 10. Ein Speicherbereich von 65.536 entspricht also 64 Kilo-Bytes. Man findet zahlreiche Kurzschreibweisen, die wie folgt lauten können:

- 64 KByte
- 64 KB
- 64K

Der Steuerbus

Um alle Aktionen, die die MPU ausführt, richtig zu koordinieren, ist eine Steuerleitung zu Synchronisation erforderlich, was die Aufgabe des Steuerbusses ist. Zu erwähnen ist hier z.B. das RD-Signal, das einen Lesezugriff auf einen bestimmten Speicherbaustein ermöglicht.

Die Taktung

Damit alle Aktionen in einer zeitlichen Abfolge von statten gehen, ist eine Zeitbasis erforderlich. Ein Taktgenerator in Zusammenarbeit mit einem externen Schwingquarz sorgt für diese Funktion. Je schneller der Quarz schwingt, desto schneller arbeitet die MPU, wobei natürlich bestimmte Spezifikationen z.B. 1 MHz oder 4 MHz eingehalten werden müssen.

Interne Register

Wir haben gesehen, dass der 6502 über den Adressbus Speicherstellen ansprechen kann. Diese Speicherstellen befinden sich entweder im ROM, RAM oder I/O. Nun besitzt der 6502 zur Durchführung diverser Aufgaben noch andere Speicherstellen, die sich im Inneren der MPU befinden. Es handelt sich um sogenannte interne Register, von denen es sechs an der Zahl gibt.

Der Programm-Zähler

Der Programm-Zähler - auch *Program Counter* - genannt, wird mit *PC* abgekürzt. Er ist ein Zähler, der immer auf die nächste auszuführende Speicherstelle zeigt und eine Datenbreite von 16 Bits besitzt.

Der Zähler wird automatisch hochgezählt, wenn ein Befehl (Instruction) abgearbeitet wurde. Er kann aber auch andere Aktionen, wie z.B. Sprungbefehle (JMP - Jump) geändert werden.

Der Stapel-Zeiger

Der Stapel-Zeiger - auch *Stack-Pointer* genannt - wird mit *SP* abgekürzt und durch einen festen Bereich innerhalb des Speichers (RAM) definiert und befindet sich im festen und nicht änderbaren Bereich zwischen $0100 und $01FF, was 256 Bytes entsprechen und besitzt eine Datenbreite von 8 Bits. Die Arbeitsweise des Stacks folgt der sogenannten *LIFO*-Struktur, was die Abkürzung für Last-In-First-Out ist. Der zuletzt auf den Stapel abgelegte Wert - ausgelöst durch einen PUSH - wird beim Abrufen - ausgelöst durch einen POP - wieder ausgegeben. Die Funktion des Stacks kommt gerade beim Aufruf von Unterprogrammen zum Tragen, wenn es darum geht, mehrere Rücksprungadressen zu speichern, die in umgekehrter Reihenfolge des Ablegens wieder abgearbeitet werden müssen. Es gibt jedoch noch weitere Einsatzgebiete wie z.B. bei der Interrupt-Verarbeitung, also Programmunterbrechung.

Der Akkumulator

Der Akkumulator ist ein Register, das die Ergebnisse der Recheneinheit - auch ALU (Arithmetic Logic Unit) genannt - aufnimmt. Wir kommen im Detail noch bei unserem ersten Beispiel darauf zu sprechen. Er besitzt eine Datenbreite von 8 Bits.

Das Index-Register X

Das Index Register X arbeitet meistens als Offset-Zähler, der zu einer Basisadresse hinzuaddiert wird, um darüber kontinuierlich einen bestimmten Speicherbereich zu adressieren. Er besitzt eine Datenbreite von 8 Bits.

Das Index-Register Y

Das Index Register Y arbeitet wie das Index-Register X meistens als Offset-Zähler, der zu einer Basisadresse hinzuaddiert wird, um darüber kontinuierlich einen bestimmten Speicherbereich zu adressieren. Er besitzt ebenfalls eine Datenbreite von 8 Bits.

Der Prozessor Status

Werden im 6502 Instruktionen abgearbeitet, haben diese ein Ergebnis, wobei dies Auswirkungen auf bestimmte Status-Flags hat. Sie werden in Abhängigkeit des Ergebnisses entweder gesetzt oder gelöscht und beziehen sich auf einzelne Bits des 8 Bit breiten Speichers.

Die Abfrage der internen Register

Nun ist es aber auf dem KIM-1 bzw. dem KIM Uno nicht so ohne weiteres möglich, sich den Inhalt der internen Register anzuschauen, wie z.B. auf einem Apple II, in dem man dort einfach den entsprechenden Befehl im Monitor Programm absetzt und auf dem Bildschirm die einzelnen Register bzw. Flags angezeigt werden. Aus diesem Grund wurde hier eine andere Möglichkeit der Abfrage geschaffen. In bestimmten Speicherstellen werden nach der Ausführung der einzelnen Befehle Kopien der Inhalte der verschiedenen internen Register angelegt. Diese lauten wie folgt:

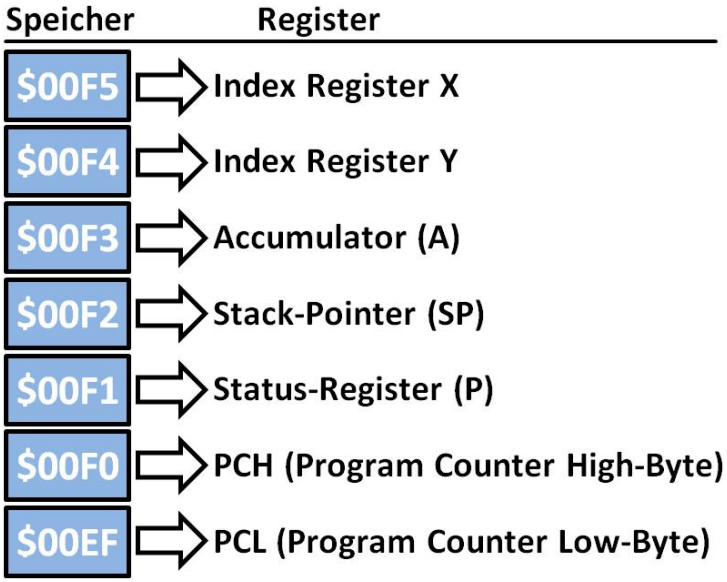


Abbildung 2: Die internen Register mit Speicheradressen

Die Speicheraufteilung

Der komplette Speicherbereich, der adressierbar ist, umfasst 64 KByte. Das haben wir schon gesehen. Wie sind aber die einzelnen Bereiche zugeordnet, denn es gibt ja ein ROM, ein RAM und den I/O-Bereich. Auf der folgenden Abbildung ist die komplette Speicheraufteilung zu sehen. Beim Kim1-Clone sind jedoch einige Abweichungen vorhanden, auf die ich noch gesondert zu sprechen komme. Schauen wir uns jedoch zunächst die originale *Memory Map* an, wie es in der Fachsprache heißt. Dabei werden bestimmte Speicherblöcke bzw. -größen zu größeren Paketen zusammengefasst, was die Übersicht steigert. Die gesamte hier gezeigte Memory Map - ohne den Erweiterungsspeicher im oberen Bereich - umfasst 8192 Bytes, was 8 KByte sind und sich von $0000 bis $1FFF erstreckt. Dieser Bereich ist in acht Unterblöcke mit der Größe von 1024 Bytes unterteilt, die mit den Bezeichnungen K0 bis K7 versehen sind. Der Buchstabe „K“ steht stellvertretend für 1 KByte. Jeder einzelne dieser K-Blöcke ist wiederum in vier Blöcke mit jeweils 256 Bytes unterteilt, die ihrerseits mit der Bezeichnung „Page“ - also Seiten - versehen sind und sich von 0 bis 31 erstrecken.

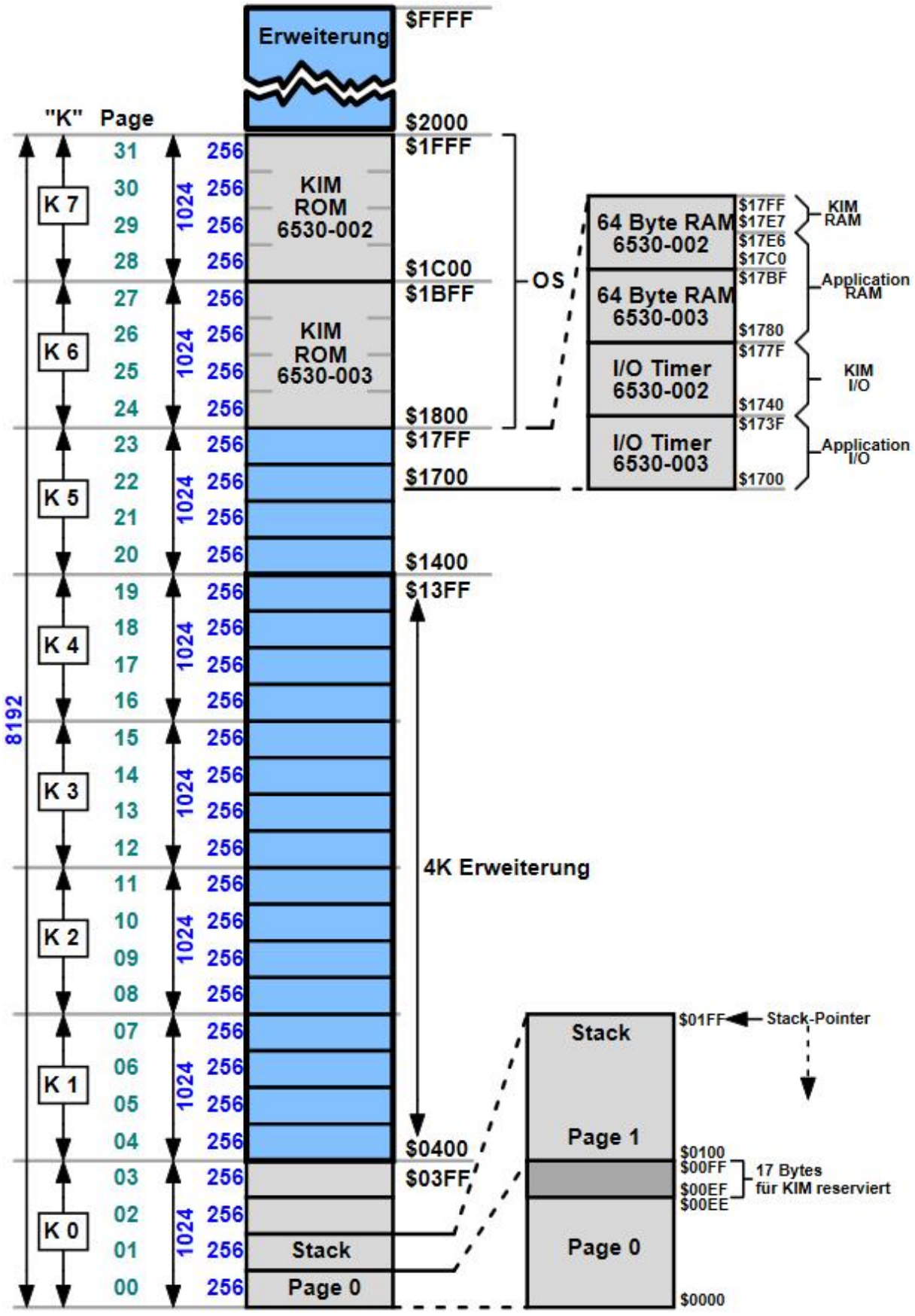
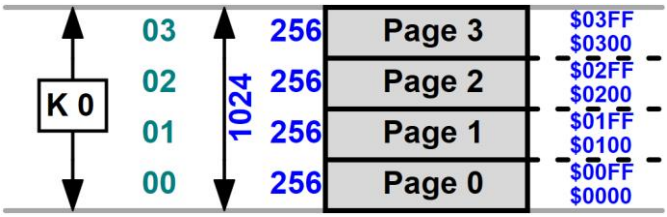


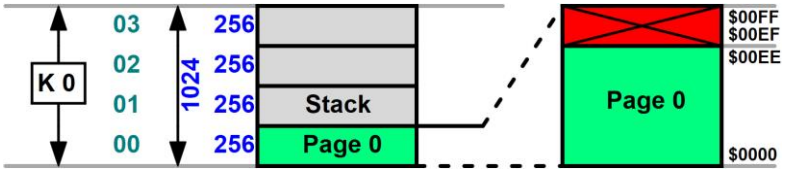
Abbildung 3: Die Speicheraufteilung

Es gilt auf jeden Fall folgendes zu beachten. Hinsichtlich des Arbeitsspeichers sollte man meinen, dass dieser Bereich vollkommen dem Benutzer für die eigenen Programme zur Verfügung steht. Das ist jedoch nicht der Fall, denn das Betriebssystem nutzt ebenfalls bestimmte Bereiche des RAMs zur Speicherung systemrelevanter Daten. Wir haben das schon bei den internen Registern gesehen, die die Adressen $00EF bis $00F5 innehaben und nicht willkürlich überschrieben werden sollten. Es ist für einen Anfänger, der sich mit der Maschinensprache noch nicht so gut auskennt, sicherlich etwas ungewohnt zu erkennen, wo Adressen beginnen, über welchen Bereich sie sich erstrecken und wo sie enden. Da kommt man schnell durcheinander und auch ich habe mich hier und da schon vertan und mich gewundert, warum mein Programm nicht funktioniert. Schauen wir uns das doch einmal exemplarisch an einem 1 KByte-Block an und sehen, wie dort die Adressen verteilt sind.

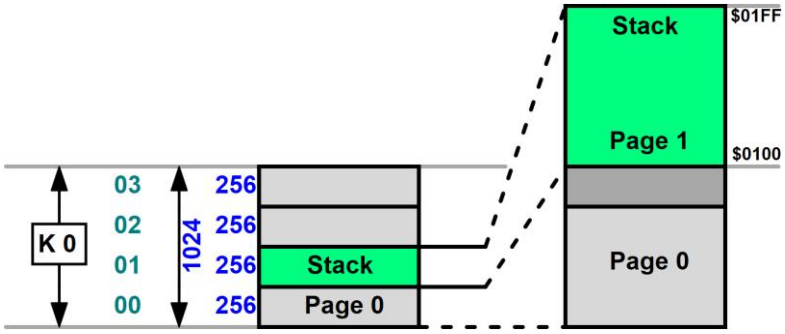


Wir erkennen hier wunderbar die Anfangs- und Endadresse jeder einzelnen Page, die aus jeweils 256 Bytes besteht. Nach dem gleichen Schema werden alle weiteren 1024-Blöcke adressiert. Die Frage, die sich uns jetzt aber stellt ist, welchen Bereich wir gefahrlos nutzen können? Ich habe diese Bereiche hier aufgelistet und farblich in grün hervorgehoben.

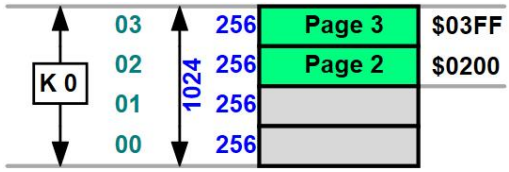
- Die gesamte Page 0 mit Ausnahme des Bereichs $00EF bis $00FF



- Die gesamte Page 1. Vorsicht, wenn der Stack genutzt wird!

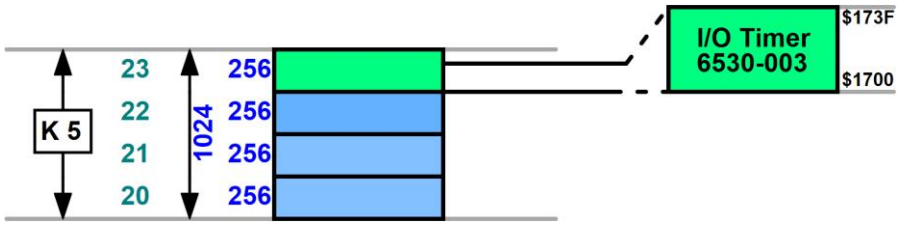


- Die gesamte Page 2 und Page 3 im Bereich von $0200 bis $03FF

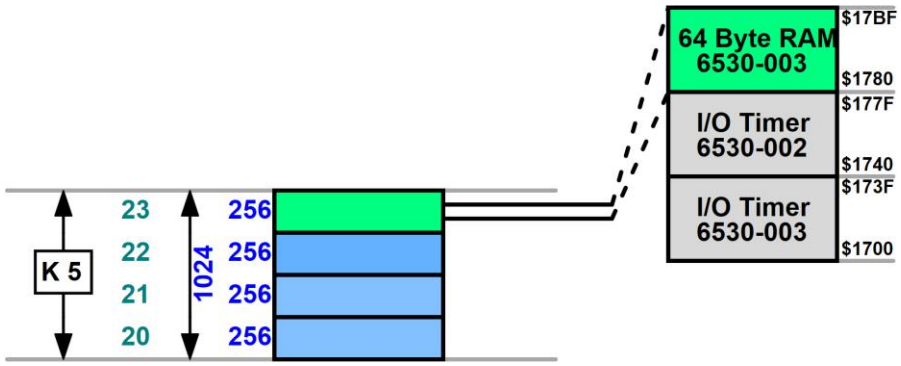


- In Page 23

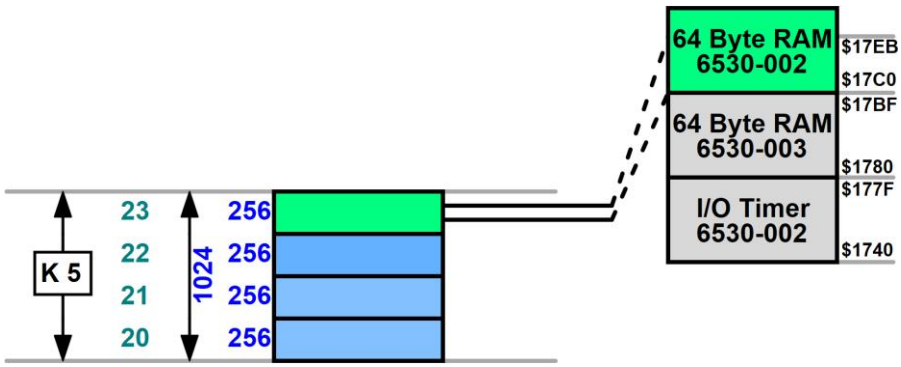
- Der I/O-Bereich zwischen $1700 und $173F



- Die gesamten 64 Bytes zwischen $1780 und $17BF



- 44 Bytes zwischen $17C0 und $17EB



Änderungen beim KIM-1 Clone

Ich hatte es schon kurz angesprochen, dass es beim KIM-1 Clone von Bob einige Abweichungen bzw. Erweiterungen in der RAM-Aufteilung gibt. Der optionale 4 KByte Erweiterungsspeicher, der von $0400 bis $13FF geht, ist beim KIM-1 Clone bestückt und kann genutzt werden. Somit erstreckt sich der durchgängige RAM-Bereich von $0000 bis $13FF. Oberhalb der Adresse $1FFF, also beginnend mit $2000 ist ebenfalls ein zusätzlicher RAM-Bereich verfügbar, der sich von $2000 bis $FFF7 erstreckt, was 56 KByte sind. In diesem Bereich können für ein EEPROM (8 KByte) die Adressen $E000 bis $FFFF genutzt werden, wobei das RAM dadurch natürlich deaktiviert wird.

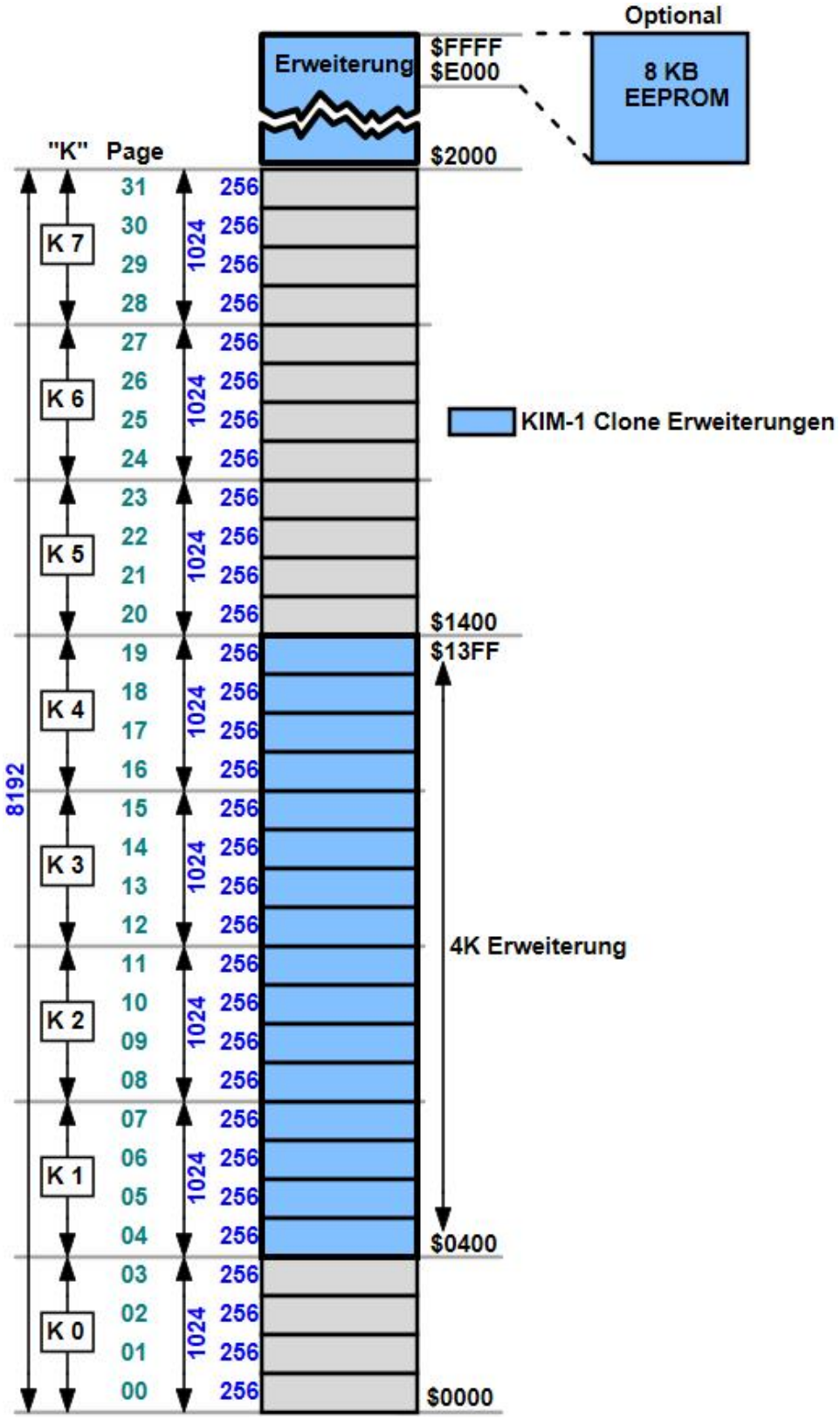


Abbildung 4: KIM-1 Clone Erweiterungen

Detailinformationen sind natürlich auf der Internetseite des KIM-1 Clone zu finden:

http://www.corshamtech.com/product/kim-clone/

Das soll als kleiner Einstieg in die Hardwareorganisation des KIM-1 genügen und wir wollen uns - wie versprochen - direkt konkreten Dingen widmen, denn es soll ja nicht allzu trocken werden.

Viel Spaß dabei...

http://erik-bartmann.de/