Headless sourcing-fähige Snippets

Hier werden kleine Helferscripte vorgeführt, die in Scripten included oder per if-then abgefragt werden können.

- mailer (Bash)
- mailer (Python3)
- fillortrim.sh (Bash)
- fillortrim.sh (Python3)
- dabch2hz.sh
- fm2hz.sh
- checkpackage.sh
- picorvid.sh
- port.sh
- rdm.sh
- rdmfreqfm.sh
- rdsps.sh
- runtxt.sh
- tvch2hzdigeu.sh

mailer (Bash)

Dieser Mailer schickt einfach eine Mail vom System aus. Dabei muss darauf geachtet werden, dass der Server Mails versenden kann mit "mail". Es schickt je nach Einstellung eine Text-Mail oder Mail mit Anhang. Der Rückgabewert besagt, ob die Mail erfolgreich versendet wurde - oder nicht. Eigener Anwendungsfall: Einfacher Text, Bestätigungscode, Link, Nachricht.

HOWTO:

```
$0 -f "/var/file.png" -e mail@domain.tld -s "Betreff"
$0 -t "Text" -m mail@domain.tld -s "Betreff"
# -e ODER -m = Mailadresse
```

Beispiel:

```
# Direktaufruf (Tests)
mailer.sh -e "mail@domain.tld" -s "Testbetreff" -t "Hallo, ich bin ein Test!"

# im Script

empfaenger="mail@domain.tld"
betreff="Testbetreff"
nachricht="Hallo, ich bin ein Test!"

if bash lib/mailer.sh -e "$empfaenger" -s "$betreff" -t "$nachricht"; then
    echo "Mail erfolgreich gesendet"
else
    echo "Fehler beim Mailversand"
fi
```

Script:

```
#!/bin/bash

while [[ $# -gt 0 ]]; do
    case "$1" in
```

```
        -f) file="$2"; shift 2 ;;
        -t) text="$2"; shift 2 ;;
        -e|-m) mail="$2"; shift 2 ;;
        -s) subj="$2"; shift 2 ;;
    esac
done

if [ -z "$mail" ] || [ -z "$subj" ]; then
    echo "ERROR"
    exit 1
fi

if [ -n "$file" ] && [ -z "$text" ]; then
    cat "$file" | mail -s "$subj" "$mail"

elif [ -n "$text" ] && [ -z "$file" ]; then
    echo "$text" | mail -s "$subj" "$mail"

else
    echo "ERROR"
    exit 1
fi

exit 0
```

mailer (Python3)

Dieser Mailer schickt einfach eine Mail vom System aus. Dabei muss darauf geachtet werden, dass der Server Mails versenden kann mit "mail". Es schickt je nach Einstellung eine Text-Mail oder Mail mit Anhang. Der Rückgabewert besagt, ob die Mail erfolgreich versendet wurde - oder nicht. Eigener Anwendungsfall: Einfacher Text, Bestätigungscode, Link, Nachricht.

HOWTO:

```
python3 $0 -f "/var/file.png" -e mail@domain.tld -s "Betreff"
python3 $0 -t "Text" -m mail@domain.tld -s "Betreff"
# -e ODER -m = Mailadresse
```

Beispiel:

```
import subprocess

empfaenger = "mail@domain.tld"
betreff = "Testbetreff"
nachricht = "Hallo, ich bin ein Test!"

try:
    result = subprocess.run(
        ['python3', 'lib/mailer.py', '-e', empfaenger, '-s', betreff, '-t', nachricht],
        check=True
    )
    print("Mail erfolgreich gesendet")
except subprocess.CalledProcessError:
    print("Fehler beim Mailversand")
```

Script:

```
#!/usr/bin/env python3

import sys
import subprocess
```

```
args = sys.argv[1:]

file = None
text = None
mail = None
subj = None

# Argument-Parsing
i = 0
while i < len(args):
    if args[i] == '-f':
        file = args[i + 1]
        i += 2
    elif args[i] == '-t':
        text = args[i + 1]
        i += 2
    elif args[i] in ['-e', '-m']:
        mail = args[i + 1]
        i += 2
    elif args[i] == '-s':
        subj = args[i + 1]
        i += 2
    else:
        i += 1

# Fehlerprüfung
if not mail or not subj:
    print("ERROR")
    sys.exit(1)

# Mailversand
if file and not text:
    try:
        subprocess.run(['mail', '-s', subj, mail], input=open(file, 'rb').read(), check=True)
    except Exception as e:
        print("Fehler beim Senden:", e)
        sys.exit(1)

elif text and not file:
```

```
    try:
        subprocess.run(['mail', '-s', subj, mail], input=text.encode(), check=True)
    except Exception as e:
        print("Fehler beim Senden:", e)
        sys.exit(1)

else:
    print("ERROR")
    sys.exit(1)
```

fillortrim.sh (Bash)

Auffüllen oder trimmen von einem Text auf X Zeichen. Es erzeugt kein typischer Rückgabewert, sondern gibt den Text bearbeitet aus.
Klassischer Anwendungsfall: textbasierte Tabelle, bei der ein Text immer die gleiche Länge haben soll, um eine sinnvolle, visuelle Tabelle zu bilden.

HOWTO:

```
$0 $var $length
$0 "Hallo ich bin ein Beispiel" 24
# immer 24 Zeichen
```

Beispiel:

```
var="Hallo ich bin ein Beispiel"
length="24"

filltext=$(bash lib/fillortrim.sh "$var" "$length")

echo "|...|$formfilltext|...|"
```

Script:

```
#!/bin/bash

var="$1"
length=${#var}
while [ $length -lt "$2" ]; do
    var="$var "
    ((length++))
done
var="${var:0:$2}"

echo "$var"
exit 0
```

fillortrim.sh (Python3)

Auffüllen oder trimmen von einem Text auf X Zeichen. Es erzeugt kein typischer Rückgabewert, sondern gibt den Text bearbeitet aus.
Klassischer Anwendungsfall: textbasierte Tabelle, bei der ein Text immer die gleiche Länge haben soll, um eine sinnvolle, visuelle Tabelle zu bilden.

HOWTO:

```
python3 $0 $var $length
python3 $0 "Hallo ich bin ein Beispiel" 24
# immer 24 Zeichen
```

Beispiel:

```
var = "Hallo ich bin ein Beispiel"
length = 24

formfilltext = var.ljust(length)[:length]

print(f"|...|{formfilltext}|...|")
```

Script:

```
import sys

def fill_or_trim(var, length):
    # Füllen bis zur gewünschten Länge
    while len(var) < length:
        var += " "
    # Abschneiden auf die gewünschte Länge
    var = var[:length]
    return var

if __name__ == "__main__":
    if len(sys.argv) != 3:
```

```
    print("Usage: python fillortrim.py <string> <length>")
    sys.exit(1)

var = sys.argv[1]
length = int(sys.argv[2])

result = fill_or_trim(var, length)
print(result)
```

dabch2hz.sh

Ausgabe von DAB-Kanal in Hertz. Ideal für Scripte, die einen Kanal übergeben und den Rückgabewert verarbeiten.
Eigener Anwendungsfall: Frequenzumrechnung in Verbindung mit einem HackRF.

Der Rückgabewert ist Hertz.

HOWTO:

```
$0 12D
```

Beispiel:

```
ch="12D"


freq=$(bash lib/dabch2hz.sh "$ch")
# Hier kann mit $freq weitergearbeitet werden
```

Script:

```
#!/bin/bash


ch=$(echo "$1" | sed 's/\([A-Z]\)/\L\1/g')
□
case "$ch" in
  5a) freq="174928000" ;;
  5b) freq="176640000" ;;
  5c) freq="178352000" ;;
  5d) freq="180064000" ;;
  6a) freq="181936000" ;;
  6b) freq="183648000" ;;
  6c) freq="185360000" ;;
  6d) freq="187072000" ;;
  7a) freq="188928000" ;;
  7b) freq="190640000" ;;
  7c) freq="192352000" ;;
```

```
7d) freq="194064000" ;;
8a) freq="195936000" ;;
8b) freq="197648000" ;;
8c) freq="199360000" ;;
8d) freq="201072000" ;;
9a) freq="202928000" ;;
9b) freq="204640000" ;;
9c) freq="206352000" ;;
9d) freq="208064000" ;;
10a) freq="209936000" ;;
10n) freq="210096000" ;;
10b) freq="211648000" ;;
10c) freq="213360000" ;;
10d) freq="215072000" ;;
11a) freq="216928000" ;;
11n) freq="217088000" ;;
11b) freq="218640000" ;;
11c) freq="220352000" ;;
11d) freq="222064000" ;;
12a) freq="223936000" ;;
12n) freq="224096000" ;;
12b) freq="225648000" ;;
12c) freq="227360000" ;;
12d) freq="229072000" ;;
13a) freq="230784000" ;;
13b) freq="232496000" ;;
13c) freq="234208000" ;;
13d) freq="235776000" ;;
13e) freq="237488000" ;;
13f) freq="239200000" ;;
la) freq="1452960000" ;;
lb) freq="1454672000" ;;
lc) freq="1456384000" ;;
ld) freq="1458096000" ;;
le) freq="1459808000" ;;
lf) freq="1461520000" ;;
lg) freq="1463232000" ;;
lh) freq="1464944000" ;;
li) freq="1466656000" ;;
lj) freq="1468368000" ;;
```

```
  lk) freq="1470080000" ;;
  ll) freq="1471792000" ;;
  lm) freq="1473504000" ;;
  ln) freq="1475216000" ;;
  lo) freq="1476928000" ;;
  lp) freq="1478640000" ;;
  2a) freq="47936000" ;;
  2b) freq="49648000" ;;
  2c) freq="51360000" ;;
  2d) freq="53072000" ;;
  3a) freq="54928000" ;;
  3b) freq="56640000" ;;
  3c) freq="58352000" ;;
  3d) freq="60064000" ;;
  4a) freq="61936000" ;;
  4b) freq="63648000" ;;
  4c) freq="65360000" ;;
  4d) freq="67072000" ;;
esac

echo "$freq"
exit 0
```

fm2hz.sh

Ausgabe einer Frequenz von FM in Hertz. Ideal für Scripte, die eine Frequenz übergeben und den Rückgabewert verarbeiten. Es ist egal, ob die Frequenz mit Punkt oder Komma geschrieben wird. Eigener Anwendungsfall: Frequenzumrechnung in Verbindung mit einem HackRF.

Der Rückgabewert ist Hertz.

HOWTO:

```
$0 89.7
```

Beispiel:

```
# Eingabe-Beispiele
eingabe="89.7"
eingabe="106,4"
eingabe="102.70"

freq=$(bash lib/fm2hz.sh "$eingabe")
# Hier kann mit $freq weitergearbeitet werden
```

Script:

```
#!/bin/bash

input=$(echo "$1" | sed 's/,/./g')

if [[ "$input" == *.* ]]; then
    integer_part=$(echo "$input" | cut -d'.' -f1)
    decimal_part=$(echo "$input" | cut -d'.' -f2)
else
    integer_part=$input
    decimal_part=""
fi

freq="${integer_part}${decimal_part}"
```

```
zeros_to_add=$((6 - ${#decimal_part}))

while [ $zeros_to_add -gt 0 ]; do
    freq="${freq}0"
    ((zeros_to_add--))
done

echo "$freq"
exit 0
```

checkpackage.sh

Es wird geprüft, ob übergebene System-Packages installiert sind. Kann auf mehreren Distros verwendet werden: Debian, Red Hat, Arch, OpenSUSE und alle Derivate davon.

Der Rückgabewert ist 0 (true) oder 1 (false).

HOWTO:

```
$0 wget
```

Beispiel:

```
if [ "$(bash lib/checkpackage.sh wget)" = "0" ]; then
    echo "installiert"
else
    echo "nicht installiert"
fi
```

Script:

```
#!/bin/bash

if [ -f /etc/debian_version ]; then
    # Debian-based
    if dpkg -s "$1" &> /dev/null; then
        echo "0"
    else
        echo "1"
    fi
elif [ -f /etc/redhat-release ]; then
    # Red Hat-based
    if rpm -q "$1" &> /dev/null; then
        echo "0"
    else
        echo "1"
```

```
    fi
elif [ -f /etc/arch-release ]; then
    # Arch-based
    if pacman -Qi "$1" &> /dev/null; then
        echo "0"
    else
        echo "1"
    fi
elif [ -f /etc/SuSE-release ]; then
    # openSUSE
    if zypper se --installed-only "$1" &> /dev/null; then
        echo "0"
    else
        echo "1"
    fi
else
    echo "Unsupported Linux distribution"
    exit 1
fi
exit 0
```

picorvid.sh

Ausgabe des Typs einer Webcam/Kamera. Ist es ein Bild oder Video - der Rückgabewert ist zur Weiterverarbeitung.
Dabei ist der Rückgabewert wie folgt:
0 = Bild
1 = Video
2 = nicht unterstützt
3 = Fehler/ungültig.
Die Weiterverarbeitung ist mit ffmpeg oder wget/curl ideal, wenn man vorher einsortieren muss, um welchen Typ es sich handelt. Bilder lassen sich wie gewohnt herunterladen und ffmpeg kann Einzelbilder aus Streams extrahieren.

HOWTO:

```
$0 URL
```

Beispiel:

```
url="http://192.168.1.10/video.cgi"
typ=$(bash lib/picorvid.sh "$url")

# VARIANTE 1
case "$typ" in
    1)
        echo "MJPEG-Stream erkannt"
        ;;
    0)
        echo "Einzelbild erkannt"
        ;;
    2)
        echo "Bekannt, aber nicht unterstützt"
        ;;
    3)
        echo "Ungültige oder unbekannte URL"
        ;;
esac

# VARIANTE 2
```

```
if [ "$typ" = "1" ]; then
    echo "Video/Stream"
elif [ "$typ" = "0" ]; then
    echo "Bild"
fi
```

Script:

```
#!/bin/bash

# Video
if echo "$1" | grep -qE
'\.mjpg|\.mjpeg|faststream|video\.cgi|GetOneShot|mjpg\.cgi|videostream\.cgi|\/image|\?action\=stream|\/cam_.\.c
gi|\.r-kom\.de'; then
echo "1"
exit 0

# Bild
elif echo "$1" | grep -qE 'snapshot\.cgi|SnapshotJPEG|\.jpg|api\.cgi|cgi-
bin\/camera|alarmimage|oneshotimage|image\/Index|CGIProxy\.fcgi|nph-
jpeg\.cgi|onvif\/snapshot|GetImage\.cgi'; then
echo "0"
exit 0

# nicht unterstützt
elif echo "$1" | grep -qE 'GetData\.cgi|mjpeg\.cgi|\.png'; then
echo "2"
exit 0
else
# ansonsten ungültig
echo "3"
exit 0
fi
```

port.sh

Gibt aus der übergebenen URL den Port aus. Falls keiner übergeben wurde, bezieht man sich auf den Standardport je nach HTTP oder HTTPS.
http://abc.de:8080/test --> 8080
https://abc.de/test --> 443
http://abc.de/test --> 80

HOWTO:

```
$0 URL
```

Beispiel:

```
url="http://beispiel.de:8080/snapshot.cgi"
port=$(bash src/port.sh "$url")
echo "Verwendeter Port: $port"
# --> 8080
```

Script:

```
#!/bin/bash

url=$(echo "$1" | sed ':a; s/\([^ ]\) /\1%20/g; ta; s/^%20//; s/%20$//' | sed s'/ //'g)

#zerlege URL in Adresse und Port
addr=$(echo "$url" | grep -oP '^https?://\K[^:/]+')
port=$(echo "$url" | grep -oP ':\K[0-9]+')

#setze Standard-Port, wenn nicht anders angegeben
if [ -z "$port" ]; then
	if echo "$url" | grep -q "https"; then
		port="443"
	else
		port="80"
	fi
fi
```

```
echo "$port"
exit 0
```

rdm.sh

Ausgabe einer random Zahl zwischen $1 und $2. Obergrenze: 32767, da RANDOM nicht mehr verarbeiten kann.

HOWTO:

```
$0 100 199
```

Beispiel:

```
min="100"
max="199"

zahl=$(bash lib/rdm.sh "$min" "$max")
```

Script:

```
#!/bin/bash

min=$1
max=$2

echo $((RANDOM % (max - min + 1) + min))
```

rdmfreqfm.sh

Es entstehen $1 random Frequenzen in einer vordefinierten Range (FM). Es kann auch ein Abstand $2 zwischen den Frequenzen eingegeben werden. Dies eignet sich gut für Frequenzplanlogistik oder Experimente.

HOWTO:

```
$0 3 5
```

Ausagabe:
88.1
107.5
97.7

Der Abstand $2 besagt, dass bei Eingabe von "2" bei einer Frequenz von 90.0 MHz NICHT 89.8, 89.9 und 90.1, 90.2 generiert werden darf.

Beispiel:

```
anzahl="3"
abstand="5"

# BEISPIEL 1
frequenzen=$(bash lib/rdmfreqfm.sh "$anzahl" "$abstand")

# BEISPIEL 2
mapfile -t freqenzen < <(bash inc/gen_frequencies.sh "$anzahl" "$abstand")

for f in "${frequenzen[@]}"; do
    echo "Frequenz: $f MHz"
done
```

Script:

```
#!/bin/bash

count="$1"
```

```
scope="$2"
min="876"
max="1079"
max_attempts="500"  # Maximale Anzahl von Versuchen, um eine nicht überlappende Zufallszahl zu finden

if [ -z "$2" ]; then
    scope="3"
fi

generated_numbers=()

function is_nearby {
    local number=$1
    for n in "${generated_numbers[@]}"; do
        if (( number >= n - $scope && number <= n + $scope )); then
            return 1
        fi
    done
    return 0
}

while [ ${#generated_numbers[@]} -lt $count ]; do
    attempts=0
    while true; do
        if (( attempts >= max_attempts )); then
            echo "ERROR"
            exit 1
        fi

        random_number=$(shuf -i ${min}-${max} -n 1)

        if is_nearby $random_number; then
            generated_numbers+=($random_number)
            formatted_number=$(echo "$random_number" | sed 's/\(.*\)\(.\)$/\1.\2/')
            echo "$formatted_number"
            break
        else
            (( attempts++ ))
        fi
    done
```

```
    attempts=0
done
```

rdsps.sh

RDS wird generiert und fix auf 8 Zeichen gesetzt. Ideal für die Weitergabe an einen RDS-Decoder. Hier kann der Name/Text an $1 übergeben werden und ein Modus für Großbuchstaben eingeschaltet werden:
$2=0 --> Großbuchstaben,
$2=1 --> normal, wie Input.
Bei kurzen Namen/Texten wird das RDS zentriert. Bei langen Namen/Texten wird das RDS gekapert auf 8 Zeichen. Untypische Zeichen fürs RDS werden ersetzt.

HOWTO:

```
input="Regio 8"
mode=1  # oder 0 für erzwungene Großbuchstaben

rds=$(bash lib/rdsps.sh "$input" "$mode")
echo "RDS-PS: [$rds]"
```

Script:

```
#!/bin/bash

if [ "$2" = "1" ]; then
    # Entferne alle Zeichen außer Großbuchstaben, Zahlen und bestimmten Satzzeichen
    var=$(echo "$1" | sed 's/[^A-Za-z0-9.,!?*-]/_/g; s/ /_/g; s/Ä/A/g; s/Ö/O/g; s/Ü/U/g; s/ä/a/g; s/ö/o/g; s/ü/u/g')
elif [ "$2" = "0" ]; then
    # Ersetze Kleinbuchstaben durch Großbuchstaben und entferne alle anderen nicht gewünschten Zeichen
    var=$(echo "$1" | sed 's/[a-z]/\U&/g' | sed 's/[^A-Z0-9.,!?*-]/_/g; s/ /_/g; s/Ä/A/g; s/Ö/O/g; s/Ü/U/g; s/ä/a/g; s/ö/o/g; s/ü/u/g')
fi

length=${#var}

if [ "$length" = "1" ]; then
    echo "___${var}____"

elif [ "$length" = "2" ]; then
```

```
    echo "___${var}___"

elif [ "$length" = "3" ]; then
    echo "__${var}___"

elif [ "$length" = "4" ]; then
    echo "__${var}__"

elif [ "$length" = "5" ]; then
    echo "_${var}__"

elif [ "$length" = "6" ]; then
    echo "_${var}_"

elif [ "$length" = "7" ]; then
    echo "${var}_"

elif [ "$length" = "8" ]; then
    echo "${var}"

elif [ "$length" -ge "9" ]; then
    var="${var:0:8}"
    echo "$var"
fi

exit 0
```

runtxt.sh

Hier kann ein Text $1 in einem textbasierten Programm durchlaufen. Hierbei lässt sich die Länge der anzuzeigenden Zeichen mit $2 bestimmen und die Schnelligkeit in ms in $3. Ideal für z. B. Displays mit begrenzter Ausgabelänge. Hier kann der Text ganz einfach durchlaufen.
$1 --> Text, der gescrollt wird
$2 --> Blockgröße (sichtbare Länge)
$3 --> Pausenzeit pro Schritt (in ms)

HOWTO:

```
$0 "Hallo Welt!" 8 100
```

Beispiel:

```
text="System läuft normal, keine Vorkommnisse"
display="16"
ms="120"

bash src/runtxt.sh "$text" "$display" "$ms"
```

Script:

```
#!/bin/bash

if [ -z "$3" ]; then
    exit 1
fi

text="$1"
block_size="$2"
sleep_time=$(echo "scale=3; $3 / 1000" | bc)

# Den Text erweitern, um eine ausreichende Anzahl von Leerzeichen für nahtloses Scrollen hinzuzufügen
padded_text="$(printf '%*s' $block_size)${text}"
padded_length=${#padded_text}
```

```
while true; do
    for (( i=0; i<padded_length; i++ )); do
        # Substring von i bis i+block_size Zeichen
        if (( i + block_size <= padded_length )); then
            substring="${padded_text:i:block_size}"
        else
            substring="${padded_text:i}"
            remaining_length=$((block_size - ${#substring}))
            substring="${substring}${padded_text:0:remaining_length}"
        fi
        echo -ne "$substring\r"
        sleep "$sleep_time"
    done
done

exit 0
```

tvch2hzdigeu.sh

Ausgabe von TV-Kanal digital in Hertz. Ideal für Scripte, die einen Kanal übergeben und den Rückgabewert verarbeiten.
Eigener Anwendungsfall: Frequenzumrechnung in Verbindung mit einem HackRF.
Dieses Snippet ist ausgelegt auf Europa, da andererorts die Frequenzen hinsichtlich der Zentralfrequenz etwas verschoben sein können.
Die Eingaben können mit Kxx, Exx, Cxx für die normalen Kanäle, Sxx für Sonderkanäle und Dxx für die Digitalkanäle (Kabelanschluss) erfolgen.
Range: K21-K69, D73-D858, S3-S41. Die Ausgabe erfolgt auf der Grundlage der Mittenfrequenz für DVB-T und DVB-C. Keine Berücksichtigung von analogen Signalen.

Hinweis: Frequenzen oberhalb von 694 MHz dürfen nicht mehr für Rundfunkausstrahlung (z. B. DVB-T) genutzt werden, da dieser Bereich durch die sogenannte digitale Dividende II für den Mobilfunk (LTE/5G) freigegeben wurde.
Die nachfolgenden Frequenzen sind daher ausschließlich aus historischen und dokumentarischen Gründen aufgeführt und dürfen nicht mehr für die Ausstrahlung verwendet werden außer in geschlossenen Systemen wie ein eigener Kabelanschluss.

HOWTO:

```
$0 K24
```

Beispiel:

```
ch="K24"


freq=$(bash lib/tvch2hzeu.sh "$ch")
# Hier kann mit $freq weitergearbeitet werden
```

Script:

```
#!/bin/bash


ch=$(echo "$1" | sed 's/\([A-Z]\)/\L\1/g' | sed -e 's/0*\([0-9]\)/\1/g')


case "$ch" in
    d73) freq="73000000" ;;
```

```
d81) freq="81000000" ;;
d114) freq="114000000" ;;
d122) freq="122000000" ;;
d130) freq="130000000" ;;
d138) freq="138000000" ;;
d146) freq="146000000" ;;
d154) freq="154000000" ;;
d162) freq="162000000" ;;
d170) freq="170000000" ;;
d178) freq="178000000" ;;
d186) freq="186000000" ;;
d194) freq="194000000" ;;
d202) freq="202000000" ;;
d210) freq="210000000" ;;
d218) freq="218000000" ;;
d226) freq="226000000" ;;
c5|k5|e5) freq="177500000" ;;
c6|k6|e6) freq="184500000" ;;
c7|k7|e7) freq="191500000" ;;
c8|k8|e8) freq="198500000" ;;
c9|k9|e9) freq="205500000" ;;
c10|k10|e10) freq="212500000" ;;
c11|k11|e11) freq="219500000" ;;
c12|k12|e12) freq="226500000" ;;
s2) freq="114000000" ;;
s3) freq="122000000" ;;
s4) freq="130000000" ;;
s6) freq="138000000" ;;
s7) freq="146000000" ;;
s8) freq="154000000" ;;
s9) freq="162000000" ;;
s10) freq="170000000" ;;
s11|d234) freq="234000000" ;;
s13|d242) freq="242000000" ;;
s14|d250) freq="250000000" ;;
s15|d258) freq="258000000" ;;
s16|d266) freq="266000000" ;;
s17|d274) freq="274000000" ;;
s18|d282) freq="282000000" ;;
s19|d290) freq="290000000" ;;
```

```
s20|d298) freq="298000000" ;;
s21) freq="306000000" ;;
s22) freq="314000000" ;;
s23) freq="322000000" ;;
s24) freq="330000000" ;;
s25) freq="338000000" ;;
s26) freq="346000000" ;;
s27) freq="354000000" ;;
s28) freq="362000000" ;;
s29) freq="370000000" ;;
s30) freq="378000000" ;;
s31) freq="386000000" ;;
s32) freq="394000000" ;;
s33) freq="402000000" ;;
s34) freq="410000000" ;;
s35) freq="418000000" ;;
s36) freq="426000000" ;;
s37) freq="434000000" ;;
s38) freq="442000000" ;;
s39) freq="450000000" ;;
s40) freq="458000000" ;;
s41) freq="466000000" ;;
c21|k21|e21|d474) freq="474000000" ;;
c22|k22|e22|d482) freq="482000000" ;;
c23|k23|e23|d490) freq="490000000" ;;
c24|k24|e24|d498) freq="498000000" ;;
c25|k25|e25|d506) freq="506000000" ;;
c26|k26|e26|d514) freq="514000000" ;;
c27|k27|e27|d522) freq="522000000" ;;
c28|k28|e28|d530) freq="530000000" ;;
c29|k29|e29|d538) freq="538000000" ;;
c30|k30|e30|d546) freq="546000000" ;;
c31|k31|e31|d554) freq="554000000" ;;
c32|k32|e32|d562) freq="562000000" ;;
c33|k33|e33|d570) freq="570000000" ;;
c34|k34|e34|d578) freq="578000000" ;;
c35|k35|e35|d586) freq="586000000" ;;
c36|k36|e36|d594) freq="594000000" ;;
c37|k37|e37|d602) freq="602000000" ;;
c38|k38|e38|d610) freq="610000000" ;;
```

```
    c39|k39|e39|d618) freq="618000000" ;;
    c40|k40|e40|d626) freq="626000000" ;;
    c41|k41|e41|d634) freq="634000000" ;;
    c42|k42|e42|d642) freq="642000000" ;;
    c43|k43|e43|d650) freq="650000000" ;;
    c44|k44|e44|d658) freq="658000000" ;;
    c45|k45|e45|d666) freq="666000000" ;;
    c46|k46|e46|d674) freq="674000000" ;;
    c47|k47|e47|d682) freq="682000000" ;;
    c48|k48|e48|d690) freq="690000000" ;;
    c49|k49|e49|d698) freq="698000000" ;;
    c50|k50|e50|d706) freq="706000000" ;;
    c51|k51|e51|d714) freq="714000000" ;;
    c52|k52|e52|d722) freq="722000000" ;;
    c53|k53|e53|d730) freq="730000000" ;;
    c54|k54|e54|d738) freq="738000000" ;;
    c55|k55|e55|d746) freq="746000000" ;;
    c56|k56|e56|d754) freq="754000000" ;;
    c57|k57|e57|d762) freq="762000000" ;;
    c58|k58|e58|d770) freq="770000000" ;;
    c59|k59|e59|d778) freq="778000000" ;;
    c60|k60|e60|d786) freq="786000000" ;;
    c61|k61|e61|d794) freq="794000000" ;;
    c62|k62|e62|d802) freq="802000000" ;;
    c63|k63|e63|d810) freq="810000000" ;;
    c64|k64|e64|d818) freq="818000000" ;;
    c65|k65|e65|d826) freq="826000000" ;;
    c66|k66|e66|d834) freq="834000000" ;;
    c67|k67|e67|d842) freq="842000000" ;;
    c68|k68|e68|d850) freq="850000000" ;;
    c69|k69|e69|d858) freq="858000000" ;;
esac

echo "$freq"
exit 0
```