

Emu6502 Reference Manual

Generated by Doxygen 1.2.17

Fri Sep 6 22:12:19 2002

Contents

1	Emu6502 Data Structure Index	1
1.1	Emu6502 Data Structures	1
2	Emu6502 File Index	3
2.1	Emu6502 File List	3
3	Emu6502 Page Index	5
3.1	Emu6502 Related Pages	5
4	Emu6502 Data Structure Documentation	7
4.1	_Instr Struct Reference	7
4.2	b4iheader Struct Reference	8
4.3	INITDLLINFO Struct Reference	9
4.4	INITDLLSTRUCT Struct Reference	10
4.5	MESSAGEDLLSTRUCT Struct Reference	12
4.6	Regs Struct Reference	13
5	Emu6502 File Documentation	15
5.1	cpu.c File Reference	15
5.2	cpu.h File Reference	41
5.3	cpudll.c File Reference	43
5.4	cpuuidll.c File Reference	48
5.5	debugdll.c File Reference	52
5.6	dllkit.c File Reference	60
5.7	dllkit.h File Reference	62
5.8	dlluser.h File Reference	66
5.9	emu6502.c File Reference	68
5.10	emu6502.h File Reference	74
5.11	genericdll.c File Reference	79

5.12	instructions.h File Reference	82
5.13	mem.c File Reference	84
5.14	mem.h File Reference	87
5.15	perifdll.c File Reference	91
5.16	usrmsgs.h File Reference	93
5.17	videodll.c File Reference	97
6	Emu6502 Page Documentation	105
6.1	Todo List	105

Chapter 1

Emu6502 Data Structure Index

1.1 Emu6502 Data Structures

Here are the data structures with brief descriptions:

_Instr (Structure of one instruction entry in the <i>INSTRUCTIONS</i> list)	7
b4iheader (DIB header)	8
INITDLLINFO (Structure containing various informations returned by <i>InitDll</i>)	9
INITDLLSTRUCT (Initial informations passed to plugin when initializing)	10
MESSAGEDLLSTRUCT (Message structure passed to plugins, when registered message occurred)	12
Regs (Registers)	13

Chapter 2

Emu6502 File Index

2.1 Emu6502 File List

Here is a list of all documented files with brief descriptions:

cpu.c	15
cpu.h	41
cpudll.c	43
cpuuidll.c	48
debugdll.c	52
dllkit.c	60
dllkit.h	62
dlluser.h	66
emu6502.c	68
emu6502.h	74
genericdll.c	79
instructions.h	82
mem.c	84
mem.h	87
perifdll.c	91
usrmsgs.h	93
videodll.c	97

Chapter 3

Emu6502 Page Index

3.1 Emu6502 Related Pages

Here is a list of all related documentation pages:

Todo List	105
-----------------	---------------------

Chapter 4

Emu6502 Data Structure Documentation

4.1 `_Instr Struct Reference`

```
#include <instructions.h>
```

4.1.1 Detailed Description

Structure of one instruction entry in the *INSTRUCTIONS* list.

Definition at line 14 of file instructions.h.

Data Fields

- char * **name**
- [AdrType](#) **aType**
- unsigned char **len**

The documentation for this struct was generated from the following file:

- [instructions.h](#)
-

4.2 b4iheader Struct Reference

4.2.1 Detailed Description

DIB header.

Definition at line 91 of file videodll.c.

Data Fields

- BITMAPV4HEADER **bmiHeader**
- RGBQUAD **bmiColors** [2]

The documentation for this struct was generated from the following file:

- [videodll.c](#)

4.3 INITDLLINFO Struct Reference

```
#include <dllkit.h>
```

4.3.1 Detailed Description

Structure containing various informations returned by *InitDll*.

Definition at line 39 of file `dllkit.h`.

Data Fields

- [MSGLIST msglist](#)
- [UINT iolo](#)
- [UINT iohi](#)

4.3.2 Field Documentation

4.3.2.1 [UINT INITDLLINFO::iohi](#)

Higher boundary of io memory reserved by this plugin

Definition at line 26 of file `dlluser.h`.

4.3.2.2 [UINT INITDLLINFO::iolo](#)

Lower boundary of io memory reserved by this plugin

Definition at line 25 of file `dlluser.h`.

4.3.2.3 [MSGLIST INITDLLINFO::msglist](#)

See also:

[MSGLIST](#)

Definition at line 24 of file `dlluser.h`.

The documentation for this struct was generated from the following files:

- [dllkit.h](#)
- [dlluser.h](#)

4.4 INITDLLSTRUCT Struct Reference

```
#include <dllkit.h>
```

4.4.1 Detailed Description

Initial informations passed to plugin when initializing.

Structure containing informations about the main application, about 6502 memory and other.

Definition at line 51 of file dllkit.h.

Data Fields

- char * [memory](#)
- HWND [hWnd](#)
- HWND [hTb](#)
- HINSTANCE [hInst](#)
- CRITICAL_SECTION [CSMemory](#)
- unsigned char(* [memgetb](#))(UINT adr)
- void(* [memsetb](#))(UINT adr, unsigned char byte)
- RECT [rcPaint](#)
- void * [optional](#)
- char * [memory](#)
- void * [optional](#)

4.4.2 Field Documentation

4.4.2.1 CRITICAL_SECTION INITDLLSTRUCT::CSMemory

Use when accesing shared [memory](#)

Definition at line 46 of file dlluser.h.

4.4.2.2 HINSTANCE INITDLLSTRUCT::hInst

Instance of the emu6502 application

Definition at line 44 of file dlluser.h.

4.4.2.3 HWND INITDLLSTRUCT::hTb

Handle of the main window toolbar

Definition at line 42 of file dlluser.h.

4.4.2.4 HWND INITDLLSTRUCT::hWnd

Handle of the main window

Definition at line 40 of file dlluser.h.

4.4.2.5 unsigned char(* INITDLLSTRUCT::memgetb)(UINT adr)

Pointer to a function which gets a byte from given adress. No need for synchronizing using [CSMemory](#).

4.4.2.6 char* INITDLLSTRUCT::memory

Pointer to shared memory. Use [CSMemory](#) when accessing it.

Definition at line 38 of file dlluser.h.

4.4.2.7 char* INITDLLSTRUCT::memory

Pointer to shared memory. Use [CSMemory](#) when accessing it.

Definition at line 53 of file dllkit.h.

4.4.2.8 void(* INITDLLSTRUCT::memsetb)(UINT adr, unsigned char byte)

Pointer to a function which sets byte on given adress. No need for synchronizing using [CSMemory](#).

4.4.2.9 void* INITDLLSTRUCT::optional

Optional data passed by emu6502.

Definition at line 56 of file dlluser.h.

4.4.2.10 void* INITDLLSTRUCT::optional

Optional data passed by emu6502.

Definition at line 71 of file dllkit.h.

4.4.2.11 RECT INITDLLSTRUCT::rcPaint

Area of allowed painting/refreshing

Definition at line 54 of file dlluser.h.

The documentation for this struct was generated from the following files:

- [dllkit.h](#)
- [dlluser.h](#)

4.5 MESSAGEDLLSTRUCT Struct Reference

```
#include <dllkit.h>
```

4.5.1 Detailed Description

Message structure passed to plugins, when registered message occurred.

Structure containing informations about message sent to plugin.

Definition at line 79 of file dllkit.h.

Data Fields

- **UINT message**
- **WPARAM wParam**
- **LPARAM lParam**

The documentation for this struct was generated from the following files:

- [dllkit.h](#)
- [dlluser.h](#)

4.6 Regs Struct Reference

```
#include <cpu.h>
```

4.6.1 Detailed Description

Registers.

Structure containing all registers of 6502

Definition at line 16 of file cpu.h.

Data Fields

- [UINT PC](#)
- [unsigned char X](#)
- [unsigned char Y](#)
- [unsigned char A](#)
- [unsigned char P](#)
- [UINT S](#)

4.6.2 Field Documentation

4.6.2.1 unsigned char Regs::A

Accumulator

Definition at line 30 of file debugdll.c.

4.6.2.2 unsigned char Regs::P

Status register

Definition at line 31 of file debugdll.c.

4.6.2.3 UINT Regs::PC

Program counter

Definition at line 27 of file debugdll.c.

4.6.2.4 UINT Regs::S

Stack pointer

Definition at line 32 of file debugdll.c.

4.6.2.5 unsigned char Regs::X

Index register X

Definition at line 28 of file debugdll.c.

4.6.2.6 unsigned char Regs::Y

Index register Y

Definition at line 29 of file debugdll.c.

The documentation for this struct was generated from the following files:

- [cpu.h](#)
- [debugdll.c](#)

Chapter 5

Emu6502 File Documentation

5.1 `cpu.c` File Reference

5.1.1 Detailed Description

The core of processor, this file contains instruction decoder and memory structure. Before use, call [Reset\(\)](#) to set correct values of registers. Execution of one instruction is done by calling [InstrStep\(\)](#).

Known bugs:

- When executing instructions in the IO area, wrong instruction timing can be computed (the IO routine might be called once for computing cycles to be consumed and once when instructions arguments are obtained.) But this bug shouldn't appear huh?

Definition in file `cpu.c`.#include "cpu.h"

Flags

- #define [FN](#) 0x80
- #define [FV](#) 0x40
- #define [FB](#) 0x10
- #define [FD](#) 0x08
- #define [FI](#) 0x04
- #define [FZ](#) 0x02
- #define [FC](#) 0x01

Adresses

- #define [RESETVKT](#) 0xFFFC
- #define [IRQVKT](#) 0xFFFE
- #define [NMIVKT](#) 0xFFFA

Flag handling defines

- #define [GETFLAG\(X\)](#) ((regs.P & (X))!=0)
-

- #define SETFLAG(X) regs.P=((regs.P) | (X))
- #define CLFLAG(X) (regs.P=regs.P & (~(X)))
- #define FLAG(BL, FL) if (BL) SETFLAG(FL); else CLFLAG(FL)
- #define SETNZFLAGS(X) FLAG((X) & 0x80, FN); FLAG(!(X), FZ)

Other defines

- #define ADR(X, Y) (((Y) << PAGESIZE)+(X))
- #define M2PC(X) regs.PC=((*memgetb)((X) + 1) << PAGESIZE) + (*memgetb)(X)
- #define HI(X) ((unsigned char) (((X) >> PAGESIZE) & 0xFF))
- #define LO(X) ((unsigned char) ((X) & 0xFF))
- #define PCSTEP(X) regs.PC+=(X)
- #define HEX2DEC(X) (((X)>>4) & 0x0F)*10 + ((X) & 0x0F)
- #define DEC2HEX(X) (((X)/10) % 10)<<4 + (X) % 10)

Stack handling defines

- #define STACKADD(X) (*memsetb)(regs.S, (X)); regs.S--; if (regs.S<0x100) regs.S=0x1FF
- #define STACKREMOVE(X) regs.S++; if (regs.S>0x1FF) regs.S=0x100; X=(*memgetb)(regs.S)

Cycle consumption defines.

- #define CONSUME(X) cycles += X
- #define CONSREL if ((oldpc & 0xFF00) != (regs.PC & 0xFF00)) CONSUME(1)
- #define CONSABSX
- #define CONSABSY
- #define CONSINDY

Memory reading/writing macros - according to addressing mode

- #define IMM regs.PC+1
- #define ZP (*memgetb)(regs.PC+1)&0xFF
- #define ZPX ((*memgetb)(regs.PC+1)+regs.X)&0xFF
- #define ZPY ((*memgetb)(regs.PC+1)+regs.Y)&0xFF
- #define ABS ADR((*memgetb)(regs.PC+1), (*memgetb)(regs.PC+2))
- #define ABSX ADR((*memgetb)(regs.PC+1), (*memgetb)(regs.PC+2))+regs.X
- #define ABSY ADR((*memgetb)(regs.PC+1), (*memgetb)(regs.PC+2))+regs.Y
- #define INDX ADR((*memgetb)((*memgetb)(regs.PC+1)+regs.X) & 0xFF), (*memgetb)((*memgetb)(regs.PC+1)+regs.X+1) & 0xFF)
- #define INDY ADR((*memgetb)((*memgetb)(regs.PC+1)), (*memgetb)((*memgetb)(regs.PC+1)+1) & 0xFF) + regs.Y
- #define REL regs.PC+1
- #define JMPADR ADR((*memgetb)(regs.PC+1), (*memgetb)(regs.PC+2))

Instruction defs

- #define [ADC](#)(MEMREAD, STEP)
- #define [AND](#)(MEMREAD, STEP)
- #define [ASL](#)(MEMREAD, STEP)
- #define [ASLA](#)
- #define [BCC](#)(MEMREAD, STEP)
- #define [BCS](#)(MEMREAD, STEP)
- #define [BEQ](#)(MEMREAD, STEP)
- #define [BNE](#)(MEMREAD, STEP)
- #define [BMI](#)(MEMREAD, STEP)
- #define [BPL](#)(MEMREAD, STEP)
- #define [BVC](#)(MEMREAD, STEP)
- #define [BVS](#)(MEMREAD, STEP)
- #define [BIT](#)(MEMREAD, STEP)
- #define [BRK](#)
- #define [CLC](#)
- #define [CLD](#)
- #define [CLI](#)
- #define [CLV](#)
- #define [CMP](#)(MEMREAD, STEP)
- #define [CPX](#)(MEMREAD, STEP)
- #define [CPY](#)(MEMREAD, STEP)
- #define [DEC](#)(MEMREAD, STEP)
- #define [DEX](#)
- #define [DEY](#)
- #define [EOR](#)(MEMREAD, STEP)
- #define [INC](#)(MEMREAD, STEP)
- #define [INX](#)
- #define [INY](#)
- #define [JMP](#)(MEMREAD) [regs.PC=MEMREAD](#)
- #define [JMPIND](#)
- #define [JSR](#)(MEMREAD)
- #define [LDA](#)(MEMREAD, STEP)
- #define [LDX](#)(MEMREAD, STEP)
- #define [LDY](#)(MEMREAD, STEP)
- #define [LSR](#)(MEMREAD, STEP)
- #define [LSRA](#)
- #define [NOP](#) PCSTEP(1)
- #define [ORA](#)(MEMREAD, STEP)
- #define [PHA](#)
- #define [PHP](#)
- #define [PLA](#)
- #define [PLP](#)
- #define [ROL](#)(MEMREAD, STEP)
- #define [ROLA](#)
- #define [ROR](#)(MEMREAD, STEP)
- #define [RORA](#)
- #define [RTI](#)
- #define [RTS](#)

- #define **SBC**(MEMREAD, STEP)
- #define **SEC**
- #define **SED**
- #define **SEI**
- #define **STA**(MEMREAD, STEP)
- #define **STX**(MEMREAD, STEP)
- #define **STY**(MEMREAD, STEP)
- #define **TAX**
- #define **TXA**
- #define **TAY**
- #define **TYA**
- #define **TSX**
- #define **TXS**

Defines

- #define **TRUE** 1
- #define **FALSE** 0
- #define **PAGESIZE** 8

Functions

- **UINT InstrStep** ()
Executes one instruction on adress stored in regs.PC.
- **Regs * GetRegs** ()
- void **SetRegs** (**Regs** r)
- void **Reset** ()
Reset interrupt.
- void **Irq** ()
IRQ interrupt.
- void **Nmi** ()
NMI interrupt.

Variables

- **Regs** regs

5.1.2 Define Documentation

5.1.2.1 #define **ABS** **ADR**((***memgetb**)(**regs.PC**+1), (***memgetb**)(**regs.PC**+2))

Absolute addressing mode

Definition at line 140 of file cpu.c.

5.1.2.2 #define ABSX ADR((*memgetb)(regs.PC+1), (*memgetb)(regs.PC+2))+regs.X

Absolute, X addressing mode

Definition at line 143 of file cpu.c.

5.1.2.3 #define ABSY ADR((*memgetb)(regs.PC+1), (*memgetb)(regs.PC+2))+regs.Y

Absolute, Y addressing mode

Definition at line 146 of file cpu.c.

5.1.2.4 #define ADC(MEMREAD, STEP)

Value:

```

if (regs.P & FD) {
    \
    tmp=HEX2DEC(regs.A)+HEX2DEC((*memgetb)(MEMREAD))+GETFLAG(FC); \
    if( tmp > 99 ) {
        \
        tmp=DEC2HEX(tmp);
        SETFLAG(FC);
        if ((tmp & 0x40)==(regs.A & 0x40)) SETFLAG(FV); \
    }
    \
    else {
        \
        tmp=DEC2HEX(tmp);
        if (!(tmp & 0x40) && (regs.A & 0x40)) SETFLAG(FV); \
    }
    \
    regs.A=tmp; \
}
\
else {
    \
    tmp=regs.A+(*memgetb)(MEMREAD)+GETFLAG(FC);
    if(tmp<regs.A) {
        \
        SETFLAG(FC);
        if ((tmp & 0x40)==(regs.A & 0x40)) SETFLAG(FV); \
    }
    \
    else {
        \
        if (!(tmp & 0x40) && (regs.A & 0x40)) SETFLAG(FV); \
    }
    \
    regs.A=tmp; \
}
\
SETNZFLAGS(regs.A); \
PCSTEP(STEP)

```

ADC: [A] + [M] + [C] -> [A], sets NVZC

Definition at line 169 of file cpu.c.

5.1.2.5 #define ADR(X, Y) (((Y) << PAGESIZE)+(X))

Converts 2 bytes to 16-bit address

Definition at line 65 of file cpu.c.

5.1.2.6 #define AND(MEMREAD, STEP)

Value:

```
regs.A=regs.A & (*memgetb)(MEMREAD); \
    SETNZFLAGS(regs.A); \
    PCSTEP(STEP)
```

AND: [A] & [M] -> [A], sets NZ

Definition at line 198 of file cpu.c.

5.1.2.7 #define ASL(MEMREAD, STEP)

Value:

```
tmp=(*memgetb)(MEMREAD); \
    FLAG(FC, tmp & 0x80); \
    tmp=(tmp << 1); \
    (*memsetb)(MEMREAD, tmp); \
    SETNZFLAGS(tmp); \
    PCSTEP(STEP)
```

ASL: [C] <- [M] <- 0, left shift, sets NZC

Definition at line 204 of file cpu.c.

5.1.2.8 #define ASLA

Value:

```
FLAG(FC, regs.A & 0x80); \
    regs.A=(regs.A << 1); \
    SETNZFLAGS(regs.A); \
    PCSTEP(1)
```

ASL: [C] <- [A] <- 0, left shift, sets NZ

Definition at line 213 of file cpu.c.

5.1.2.9 #define BCC(MEMREAD, STEP)

Value:

```
if (!GETFLAG(FC)) { \
    CONSUME(3); \
    tmp=(*memgetb)(MEMREAD); \
    if (tmp>0x7F) { \
        PCSTEP(STEP-0x100+tmp); \
    } \
    else { \
        PCSTEP(STEP+tmp); \
    } \
} \
else { \
    PCSTEP(STEP); \
    CONSUME(2); \
}
```

BCC: relative jump, if [C]=0

Definition at line 220 of file cpu.c.

5.1.2.10 #define BCS(MEMREAD, STEP)**Value:**

```

if (GETFLAG(FC)) {
    CONSUME(3);
    tmp=(*memgetb)(MEMREAD);
    if (tmp>0x7F) {
        PCSTEP(STEP-0x100+tmp);
    }
    else {
        PCSTEP(STEP+tmp);
    }
}
else {
    PCSTEP(STEP);
    CONSUME(2);
}

```

BCS: relative jump, if [C]=1

Definition at line 237 of file cpu.c.

5.1.2.11 #define BEQ(MEMREAD, STEP)**Value:**

```

if (GETFLAG(FZ)) {
    CONSUME(3);
    tmp=(*memgetb)(MEMREAD);
    if (tmp>0x7F) {
        PCSTEP(STEP-0x100+tmp);
    }
    else {
        PCSTEP(STEP+tmp);
    }
}
else {
    PCSTEP(STEP);
    CONSUME(2);
}

```

BEQ: relative jump, if [Z]=1

Definition at line 254 of file cpu.c.

5.1.2.12 #define BIT(MEMREAD, STEP)**Value:**

```

tmp=(*memgetb)(MEMREAD);
FLAG(FN, tmp & 0x80);
FLAG(FV, tmp & 0x40);
tmp=tmp & regs.A;
FLAG(FZ, ~tmp);
PCSTEP(STEP)

```

BIT: [A] and [M], [M7] -> [N], [M6] -> [V], result sets [Z]

Definition at line 356 of file cpu.c.

5.1.2.13 #define BMI(MEMREAD, STEP)**Value:**

```

if (GETFLAG(FN)) {
    CONSUME(3);
    tmp=(*memgetb)(MEMREAD);
    if (tmp>0x7F) {
        PCSTEP(STEP-0x100+tmp);
    }
    else {
        PCSTEP(STEP+tmp);
    }
}
else {
    PCSTEP(STEP);
    CONSUME(2);
}

```

BMI: relative jump, if [N]=1

Definition at line 288 of file cpu.c.

5.1.2.14 #define BNE(MEMREAD, STEP)**Value:**

```

if (!GETFLAG(FZ)) {
    CONSUME(3);
    tmp=(*memgetb)(MEMREAD);
    if (tmp>0x7F) {
        PCSTEP(STEP-0x100+tmp);
    }
    else {
        PCSTEP(STEP+tmp);
    }
}
else {
    PCSTEP(STEP);
    CONSUME(2);
}

```

BNE: relative jump, if [Z]=0

Definition at line 271 of file cpu.c.

5.1.2.15 #define BPL(MEMREAD, STEP)**Value:**

```

if (!GETFLAG(FN)) {
    CONSUME(3);
    tmp=(*memgetb)(MEMREAD);
    if (tmp>0x7F) {
        PCSTEP(STEP-0x100+tmp);
    }
    else {
        PCSTEP(STEP+tmp);
    }
}

```



```

if (GETFLAG(FC)) {
    CONSUME(3);
    tmp=(*memgetb)(MEMREAD);
    if (tmp>0x7F) {
        PCSTEP(STEP-0x100+tmp);
    }
    else {
        PCSTEP(STEP+tmp);
    }
}
else {
    PCSTEP(STEP);
    CONSUME(2);
}

```

BVS: relative jump, if [V]=1

Definition at line 339 of file cpu.c.

5.1.2.19 #define CLC

Value:

```

CLFLAG(FC);
PCSTEP(1)

```

CLC: 0 -> [C]

Definition at line 378 of file cpu.c.

5.1.2.20 #define CLD

Value:

```

CLFLAG(FD);
PCSTEP(1)

```

CLD: 0 -> [D]

Definition at line 383 of file cpu.c.

5.1.2.21 #define CLFLAG(X) (regs.P=regs.P & (~(X)))

Sets flag X to 0. Affects P register

Definition at line 52 of file cpu.c.

5.1.2.22 #define CLI

Value:

```

CLFLAG(FI);
PCSTEP(1)

```

CLI: 0 -> [I]

Definition at line 388 of file cpu.c.

5.1.2.23 #define CLV**Value:**

```
CLFLAG(FV);          \
    PCSTEP(1)
```

CLV: 0 -> [V]

Definition at line 393 of file cpu.c.

5.1.2.24 #define CMP(MEMREAD, STEP)**Value:**

```
tmp=(*memgetb)(MEMREAD);          \
    if (regs.A<tmp) SETFLAG(FC);    \
    SETNZFLAGS(regs.A-tmp);        \
    PCSTEP(STEP)
```

CMP: [A] - [M], result sets NZC

Definition at line 398 of file cpu.c.

5.1.2.25 #define CONSABSX**Value:**

```
if ((ADR((*memgetb)(regs.PC+1), (*memgetb)(regs.PC+2)) & 0xFF00) !=          \
    ((ADR((*memgetb)(regs.PC+1), (*memgetb)(regs.PC+2)) + regs.X) & 0xFF00)) \
    CONSUME(1)
```

If page boundary is crossed, add 1 cycle

Definition at line 109 of file cpu.c.

5.1.2.26 #define CONSABSY**Value:**

```
if ((ADR((*memgetb)(regs.PC+1), (*memgetb)(regs.PC+2)) & 0xFF00) !=          \
    ((ADR((*memgetb)(regs.PC+1), (*memgetb)(regs.PC+2)) + regs.Y) & 0xFF00)) \
    CONSUME(1)
```

If page boundary is crossed, add 1 cycle

Definition at line 114 of file cpu.c.

5.1.2.27 #define CONSINDY**Value:**

```

if ((ADR(*memgetb)(*memgetb(regs.PC+1)), (*memgetb)(*memgetb(regs.PC+1))+1) & 0xFF00) != \
    ((ADR(*memgetb)(*memgetb(regs.PC+1)), (*memgetb)(*memgetb(regs.PC+1))+1) + regs.Y) &
    CONSUME(1)

```

If page boundary is crossed, add 1 cycle

Definition at line 119 of file cpu.c.

5.1.2.28 #define CONSREL if ((oldpc & 0xFF00) != (regs.PC & 0xFF00)) CONSUME(1)

If branch crosses a page boundary, add 1 cycle

Definition at line 106 of file cpu.c.

5.1.2.29 #define CONSUME(X) cycles += X

Consume X cycles

Definition at line 103 of file cpu.c.

5.1.2.30 #define CPX(MEMREAD, STEP)

Value:

```

tmp=(*memgetb)(MEMREAD); \
    if (regs.X<tmp) SETFLAG(FC); \
    SETNZFLAGS(regs.X-tmp); \
    PCSTEP(STEP)

```

CPX: [X] - [M], result sets NZC

Definition at line 405 of file cpu.c.

5.1.2.31 #define CPY(MEMREAD, STEP)

Value:

```

tmp=(*memgetb)(MEMREAD); \
    if (regs.Y<tmp) SETFLAG(FC); \
    SETNZFLAGS(regs.Y-tmp); \
    PCSTEP(STEP)

```

CPY: [Y] - [M], result sets NZC

Definition at line 412 of file cpu.c.

5.1.2.32 #define DEC(MEMREAD, STEP)

Value:

```

tmp=(*memgetb)(MEMREAD); \
    tmp--; \
    SETNZFLAGS(tmp); \
    (*memsetb)(MEMREAD, tmp); \
    PCSTEP(STEP)

```

DEC: [M] - 1 -> [M], sets NZ

Definition at line 419 of file cpu.c.

5.1.2.33 #define DEC2HEX(X) (((((X)/10) % 10) << 4) + (X) % 10)

Converts number in BCD code to std. decimal. Example> \$10 hex -> 10 dec

Definition at line 83 of file cpu.c.

5.1.2.34 #define DEX

Value:

```
regs.X--;          \
    SETNZFLAGS(regs.X);    \
    PCSTEP(1)
```

DEX: [X] -1 -> [X], sets NZ

Definition at line 427 of file cpu.c.

5.1.2.35 #define DEY

Value:

```
regs.Y--;          \
    SETNZFLAGS(regs.Y);    \
    PCSTEP(1)
```

DEY: [Y] - 1 -> [Y], sets NZ

Definition at line 433 of file cpu.c.

5.1.2.36 #define EOR(MEMREAD, STEP)

Value:

```
regs.A=regs.A ^ (*memgetb)(MEMREAD);    \
    SETNZFLAGS(regs.A);    \
    PCSTEP(STEP)
```

EOR: [A] xor [M] -> [A], sets NZ

Definition at line 439 of file cpu.c.

5.1.2.37 #define FB 0x10

BRK flag

Definition at line 28 of file cpu.c.

5.1.2.38 #define FC 0x01

Carry flag

Definition at line 32 of file cpu.c.

5.1.2.39 #define FD 0x08

Decimal flag

Definition at line 29 of file cpu.c.

5.1.2.40 #define FI 0x04

Interrupt flag

Definition at line 30 of file cpu.c.

5.1.2.41 #define FLAG(BL, FL) if (BL) SETFLAG(FL); else CLFLAG(FL)

If condition BL is satisfied, flag FL is set to 1 otherwise 0

Definition at line 55 of file cpu.c.

5.1.2.42 #define FN 0x80

Negative flag

Definition at line 26 of file cpu.c.

5.1.2.43 #define FV 0x40

Overflow flag

Definition at line 27 of file cpu.c.

5.1.2.44 #define FZ 0x02

Zero flag

Definition at line 31 of file cpu.c.

5.1.2.45 #define GETFLAG(X) ((regs.P & (X))!=0)

Gets value of flag X, for conditions

Definition at line 46 of file cpu.c.

5.1.2.46 #define HEX2DEC(X) (((X)>>4) & 0x0F)*10 + ((X) & 0x0F)

Converts number to BCD. Example: 10 dec -> \$10 hex

Definition at line 80 of file cpu.c.

5.1.2.47 #define HI(X) ((unsigned char) (((X) >> PAGESIZE) & 0xFF))

Higher byte of word

Definition at line 71 of file cpu.c.

5.1.2.48 #define IMM regs.PC+1

Immediate addressing mode

Definition at line 128 of file cpu.c.

5.1.2.49 #define INC(MEMREAD, STEP)

Value:

```
tmp = (*memgetb)(MEMREAD);          \
    tmp++;                          \
    SETNZFLAGS(tmp);                \
    (*memsetb)(MEMREAD, tmp);       \
    PCSTEP(STEP)
```

INC: [M] + 1 -> [M], sets NZ

Definition at line 445 of file cpu.c.

5.1.2.50 #define INDX ADDR((*memgetb)((*memgetb)(regs.PC+1)+regs.X) & 0xFF), (*memgetb)((*memgetb)(regs.PC+1)+regs.X+1) & 0xFF)

Indirect, X addressing mode

Definition at line 149 of file cpu.c.

5.1.2.51 #define INDY ADDR((*memgetb)((*memgetb)(regs.PC+1)), (*memgetb)((*memgetb)(regs.PC+1)+1) & 0xFF) + regs.Y

Indirect, Y addressing mode

Definition at line 152 of file cpu.c.

5.1.2.52 #define INX

Value:

```
regs.X++;                          \
    SETNZFLAGS(regs.X);            \
    PCSTEP(1)
```

INX: [X] + 1 -> [X], sets NZ

Definition at line 453 of file cpu.c.

5.1.2.53 #define INY**Value:**

```
regs.Y++; \
    SETNZFLAGS(regs.Y); \
    PCSTEP(1)
```

INY: [Y] + 1 -> [Y], sets NZ

Definition at line 459 of file cpu.c.

5.1.2.54 #define IRQVKT 0xFFFE

IRQ jumps here

Definition at line 39 of file cpu.c.

5.1.2.55 #define JMP(MEMREAD) regs.PC=MEMREAD

JMP: [M] -> [PC] JMP(..) only for ABSOLUTE addressing! Explanation follows

Definition at line 467 of file cpu.c.

5.1.2.56 #define JMPADR ADR((*memgetb)(regs.PC+1), (*memgetb)(regs.PC+2))

Used only for jmp, should be absolute

Definition at line 158 of file cpu.c.

5.1.2.57 #define JMPIND**Value:**

```
regs.PC=ADR( \
    (*memgetb)(ADR((*memgetb)(regs.PC+1), (*memgetb)(regs.PC+2))), \
    (*memgetb)(ADR((*memgetb)(regs.PC+1)+1) & 0xFF, (*memgetb)(regs.PC+2)))
```

Indirect jump must never use address beginning on 0xFFFF. if so, jump will be done to address stored in 0xFFFF and 0XXX00 ! (no page overflow)

Definition at line 473 of file cpu.c.

5.1.2.58 #define JSR(MEMREAD)**Value:**

```
tmpjsr=MEMREAD; \
    PCSTEP(2); \
    STACKADD(HI(regs.PC)); \
    STACKADD(LO(regs.PC)); \
    regs.PC=tmpjsr
```

JSR: [PC] + 2 - 1 -> Stack, [M] -> PC JSR puts address of next instruction-1 on the stack (rts adds +1)

Definition at line 481 of file cpu.c.

5.1.2.59 #define LDA(MEMREAD, STEP)**Value:**

```
regs.A=(*memgetb)(MEMREAD);    \
        SETNZFLAGS(regs.A);    \
        PCSTEP(STEP)
```

LDA: [M] -> [A], sets NZ

Definition at line 489 of file cpu.c.

5.1.2.60 #define LDX(MEMREAD, STEP)**Value:**

```
regs.X=(*memgetb)(MEMREAD);    \
        SETNZFLAGS(regs.X);    \
        PCSTEP(STEP)
```

LDX: [M] -> [X], sets NZ

Definition at line 495 of file cpu.c.

5.1.2.61 #define LDY(MEMREAD, STEP)**Value:**

```
regs.Y=(*memgetb)(MEMREAD);    \
        SETNZFLAGS(regs.Y);    \
        PCSTEP(STEP)
```

LDY: [M] -> [Y], sets NZ

Definition at line 501 of file cpu.c.

5.1.2.62 #define LO(X) ((unsigned char) ((X) & 0xFF))

Lower byte of word

Definition at line 74 of file cpu.c.

5.1.2.63 #define LSR(MEMREAD, STEP)**Value:**

```
tmp=(*memgetb)(MEMREAD);      \
     FLAG(FC, tmp & 0x01);     \
     tmp=(tmp >> 1);          \
     (*memsetb)(MEMREAD, tmp); \
     SETNZFLAGS(tmp);         \
     PCSTEP(STEP)
```

LSR: 0 -> [M] -> [C], right shift, sets: NZC

Definition at line 507 of file cpu.c.

5.1.2.64 #define LSRA**Value:**

```
FLAG(FC, regs.A & 0x01);      \
regs.A=(regs.A >> 1);        \
SETNZFLAGS(regs.A);          \
PCSTEP(1)
```

LSR: 0 -> [A] -> [C], right shift, sets: NZC

Definition at line 516 of file cpu.c.

5.1.2.65 #define M2PC(X) regs.PC=((*memgetb)((X) + 1) << PAGESIZE) + (*memgetb)(X)

Sets PC to 2b address stored in memory from adress X

Definition at line 68 of file cpu.c.

5.1.2.66 #define NOP PCSTEP(1)

NOP: Reserved

Definition at line 523 of file cpu.c.

5.1.2.67 #define ORA(MEMREAD, STEP)**Value:**

```
regs.A=regs.A | (*memgetb)(MEMREAD); \
SETNZFLAGS(regs.A);                 \
PCSTEP(STEP)
```

ORA: [A] or [M] -> [A], sets NZ

Definition at line 527 of file cpu.c.

5.1.2.68 #define PCSTEP(X) regs.PC+=X

Moves execution point

Definition at line 77 of file cpu.c.

5.1.2.69 #define PHA**Value:**

```
STACKADD(regs.A);              \
PCSTEP(1)
```

PHA: A -> Stack

Definition at line 533 of file cpu.c.

5.1.2.70 #define PHP**Value:**

```
STACKADD(regs.P);      \
    PCSTEP(1)
```

PHP: P -> Stack

Definition at line 538 of file cpu.c.

5.1.2.71 #define PLA**Value:**

```
STACKREMOVE(regs.A);  \
    SETNZFLAGS(regs.A); \
    PCSTEP(1)
```

PLA: Stack -> A, sets NZ

Definition at line 543 of file cpu.c.

5.1.2.72 #define PLP**Value:**

```
STACKREMOVE(regs.P);  \
    regs.P=regs.P | 0x20; \
    PCSTEP(1)
```

PLP: Stack -> P, sets NVBDIZC

Definition at line 549 of file cpu.c.

5.1.2.73 #define REL [regs.PC+1](#)

Relative addressing mode

Definition at line 155 of file cpu.c.

5.1.2.74 #define RESETVKT 0xFFFC

Reset interrupt jumps here

Definition at line 38 of file cpu.c.

5.1.2.75 #define ROL(MEMREAD, STEP)**Value:**

```

tmp=( *memgetb)(MEMREAD);          \
    tmpc=GETFLAG(FC);              \
    FLAG(FC, tmp & 0x80);          \
    tmp=(tmp << 1) | ((tmpc) ? 0x01 : 0x00); \
    (*memsetb)(MEMREAD, tmp);      \
    PCSTEP(STEP)

```

ROL: [C_new] <- [M] <- [C_old], left rotation, sets NZC

Definition at line 555 of file cpu.c.

5.1.2.76 #define ROLA

Value:

```

tmpc=GETFLAG(FC);          \
    FLAG(FC, regs.A & 0x80); \
    regs.A=(regs.A << 1) | ((tmpc) ? 0x01 : 0x00); \
    PCSTEP(1)

```

ROL: [C_new] <- [A] <- [C_old], left rotation, sets NZC

Definition at line 564 of file cpu.c.

5.1.2.77 #define ROR(MEMREAD, STEP)

Value:

```

tmp=( *memgetb)(MEMREAD);          \
    tmpc=GETFLAG(FC);              \
    FLAG(FC, tmp & 0x01);          \
    tmp=(tmp >> 1) | ((tmpc) ? 0x80 : 0x00); \
    (*memsetb)(MEMREAD, tmp);      \
    PCSTEP(STEP)

```

ROR: [C_old] -> [M] -> [C_new], right rotation, sets NZC

Definition at line 571 of file cpu.c.

5.1.2.78 #define RORA

Value:

```

tmpc=GETFLAG(FC);          \
    FLAG(FC, regs.A & 0x01); \
    regs.A=(regs.A >> 1) | ((tmpc) ? 0x80 : 0x00); \
    PCSTEP(1)

```

ROR A: [C_old] -> [A] -> [C_new], right rotation, sets NZC

Definition at line 580 of file cpu.c.

5.1.2.79 #define RTI**Value:**

```
STACKREMOVE(regs.P); \
    STACKREMOVE(tmp); \
    STACKREMOVE(regs.PC); \
    regs.PC=(regs.PC << 8) + tmp
```

RTI: Stack -> [P], Stack -> [PC], return from interrupt, sets NVBDIZC

Definition at line 587 of file cpu.c.

5.1.2.80 #define RTS**Value:**

```
STACKREMOVE(tmp); \
    STACKREMOVE(regs.PC); \
    regs.PC=(regs.PC << 8) + tmp + 1
```

RTS: Stack + 1 -> [PC], return from JSR subroutine (for +1 see JSR)

Definition at line 594 of file cpu.c.

5.1.2.81 #define SBC(MEMREAD, STEP)**Value:**

```
if (regs.P & FD) { \
    tmp=HEX2DEC(regs.A)-HEX2DEC((*memgetb)(MEMREAD)-(1-GETFLAG(FC))); \
    if (tmp<=99) { \
        tmp=DEC2HEX(tmp); \
        SETFLAG(FC); \
        if ((tmp & 0x40) && !(regs.A & 0x40)) SETFLAG(FV); \
    } \
    else { \
        tmp=tmp-156; \
        tmp=DEC2HEX(tmp); \
        if ((tmp & 0x40)==(regs.A & 0x40)) SETFLAG(FV); \
    } \
    regs.A=tmp; \
} \
else { \
    tmp=regs.A-(*memgetb)(MEMREAD)-(1-GETFLAG(FC)); \
    if (tmp<=regs.A) { \
        SETFLAG(FC); \
        if ((tmp & 0x40) && !(regs.A & 0x40)) SETFLAG(FV); \
    } \
    else { \
        if ((tmp & 0x40)==(regs.A & 0x40)) SETFLAG(FV); \
    } \
    regs.A=tmp; \
} \
SETNZFLAGS(regs.A); \
PCSTEP(STEP)
```

SBC: [A] - [M] - non[C] -> [A], sets NVZ(non C)

Definition at line 600 of file cpu.c.

5.1.2.82 #define SEC**Value:**

```
SETFLAG(FC);    \
    PCSTEP(1)
```

SEC: 1 -> [C]

Definition at line 630 of file cpu.c.

5.1.2.83 #define SED**Value:**

```
SETFLAG(FD);    \
    PCSTEP(1)
```

SED: 1 -> [D]

Definition at line 635 of file cpu.c.

5.1.2.84 #define SEI**Value:**

```
SETFLAG(FI);    \
    PCSTEP(1)
```

SEI: 1 -> [I]

Definition at line 640 of file cpu.c.

5.1.2.85 #define SETFLAG(X) regs.P=((regs.P) | (X))

Sets flag X to 1. Affects P register

Definition at line 49 of file cpu.c.

5.1.2.86 #define SETNZFLAGS(X) FLAG((X) & 0x80, FN); FLAG(!(X), FZ)

Sets N and Z flag according to X

Definition at line 58 of file cpu.c.

5.1.2.87 #define STA(MEMREAD, STEP)**Value:**

```
(*memsetb)(MEMREAD, regs.A);    \
    PCSTEP(STEP)
```

STA: [A] -> [M]

Definition at line 645 of file cpu.c.

5.1.2.88 #define STACKADD(X) (*memsetb)(regs.S, (X)); regs.S-; if (regs.S<0x100) regs.S=0x1FF

Adds byte X to/from the top of stack

Definition at line 91 of file cpu.c.

5.1.2.89 #define STACKREMOVE(X) regs.S++; if (regs.S>0x1FF) regs.S=0x100; X=(*memgetb)(regs.S)

Removes byte X to/from the top of stack

Definition at line 94 of file cpu.c.

5.1.2.90 #define STX(MEMREAD, STEP)**Value:**

```
(*memsetb)(MEMREAD, regs.X); \
    PCSTEP(STEP)
```

STX: [X] -> [M]

Definition at line 650 of file cpu.c.

5.1.2.91 #define STY(MEMREAD, STEP)**Value:**

```
(*memsetb)(MEMREAD, regs.Y); \
    PCSTEP(STEP)
```

STY: [Y] -> [M]

Definition at line 655 of file cpu.c.

5.1.2.92 #define TAX**Value:**

```
regs.X=regs.A; \
    SETNZFLAGS(regs.X); \
    PCSTEP(1)
```

TAX: [A] -> [X], sets NZ

Definition at line 660 of file cpu.c.

5.1.2.93 #define TAY**Value:**

```
regs.Y=regs.A; \
    SETNZFLAGS(regs.Y); \
    PCSTEP(1)
```

TAY: [A] -> [Y], sets NZ

Definition at line 672 of file cpu.c.

5.1.2.94 #define TSX

Value:

```
regs.X=regs.S; \
    SETNZFLAGS(regs.X); \
    PCSTEP(1)
```

TSX: [S] -> [X], sets NZ

Definition at line 684 of file cpu.c.

5.1.2.95 #define TXA

Value:

```
regs.A=regs.X; \
    SETNZFLAGS(regs.A); \
    PCSTEP(1)
```

TXA: [X] -> [A], sets NZ

Definition at line 666 of file cpu.c.

5.1.2.96 #define TXS

Value:

```
regs.S=regs.X; \
    PCSTEP(1)
```

TXS: [X] -> [S]

Definition at line 690 of file cpu.c.

5.1.2.97 #define TYA

Value:

```
regs.A=regs.Y; \
    SETNZFLAGS(regs.A); \
    PCSTEP(1)
```

TYA: [Y] -> [A], sets NZ

Definition at line 678 of file cpu.c.

5.1.2.98 #define ZP (*memgetb)(regs.PC+1)&0xFF

Zero page addressing mode

Definition at line 131 of file cpu.c.

5.1.2.99 #define ZPX ((*memgetb)(regs.PC+1)+regs.X)&0xFF

Zero page, X addressing mode

Definition at line 134 of file cpu.c.

5.1.2.100 #define ZPY ((*memgetb)(regs.PC+1)+regs.Y)&0xFF

Zero page, Y addressing mode

Definition at line 137 of file cpu.c.

5.1.3 Function Documentation**5.1.3.1 UINT InstrStep ()**

Executes one instruction on adress stored in *regs.PC*.

Executes one instruction on adress stored in *regs.PC*; increments (or changes in case of jump) program counter

Returns :

Number of cycles consumed

Todo:

Add undocumented instructions

Definition at line 706 of file cpu.c.

5.1.3.2 void Irq ()

IRQ interrupt.

Adds actual position and the P register (flags) to the top of the stack, sets the I (interrupt) flag to 1 and jumps to the *IRQVKT*, reset initialization routine.

Definition at line 1379 of file cpu.c.

5.1.3.3 void Nmi ()

NMI interrupt.

Adds actual position and the P register (flags) to the top of the stack, sets the I (interrupt) flag to 1 and jumps to the *NMIVKT*, reset initialization routine.

Definition at line 1398 of file cpu.c.

5.1.3.4 void Reset ()

Reset interrupt.

Zeroes A, X, Y registers, resets stack pointer S, initializes P register (zeroes all flags except I, which is set to 1) and jumps to the *RESETVKT*, reset initialization routine.

Definition at line 1362 of file cpu.c.

5.1.4 Variable Documentation

5.1.4.1 **Regs** regs

Variable containing all registers

Definition at line 696 of file cpu.c.

5.2 cpu.h File Reference

5.2.1 Detailed Description

Header file for [cpu.c](#)

Definition in file [cpu.h](#).

Data Structures

- struct [Regs](#)
Registers.

Typedefs

- typedef unsigned int [UINT](#)

Functions

- void [Reset](#) ()
Reset interrupt.
- void [Irq](#) ()
IRQ interrupt.
- void [Nmi](#) ()
NMI interrupt.
- [UINT InstrStep](#) ()
Executes one instruction on adress stored in regs.PC.
- [Regs * GetRegs](#) ()
- void [SetRegs](#) ([Regs r](#))

Variables

- unsigned char(* [memgetb](#))(UINT adr)
- void(* [memsetb](#))(UINT adr, unsigned char byte)

5.2.2 Function Documentation

5.2.2.1 [UINT InstrStep](#) ()

Executes one instruction on adress stored in *regs.PC*.

Executes one instruction on adress stored in *regs.PC*; increments (or changes in case of jump) program counter

Returns :

Number of cycles consumed

Todo:

Add undocumented instructions

Definition at line 706 of file cpu.c.

5.2.2.2 void Irq ()

IRQ interrupt.

Adds actual position and the P register (flags) to the top of the stack, sets the I (interrupt) flag to 1 and jumps to the *IRQVKT*, reset initialization routine.

Definition at line 1379 of file cpu.c.

5.2.2.3 void Nmi ()

NMI interrupt.

Adds actual position and the P register (flags) to the top of the stack, sets the I (interrupt) flag to 1 and jumps to the *NMIVKT*, reset initialization routine.

Definition at line 1398 of file cpu.c.

5.2.2.4 void Reset ()

Reset interrupt.

Zeroes A, X, Y registers, resets stack pointer S, initializes P register (zeroes all flags except I, which is set to 1) and jumps to the *RESETVKT*, reset initialization routine.

Definition at line 1362 of file cpu.c.

5.3 cpudll.c File Reference

5.3.1 Detailed Description

This dll creates a cpu thread in which cpu is running. Also handles correct cpu timing and functionality of interrupt mechanism.

Definition in file `cpudll.c`:

```
#include <windows.h>
```

```
#include <stdlib.h>
```

```
#include <tchar.h>
```

```
#include "../dllkit/dllkit.h"
```

```
#include "cpu.h"
```

```
#include "../usrmsgs.h"
```

Enumerations

- enum **THRSTATE** { **NOTSTARTED**, **RUNNING**, **ENDED** }
- enum **INTR** { **NONE**, **IRQ**, **NMI**, **RESET** }

Functions

- DWORD WINAPI **CpuThreadFunc** (LPVOID lpParam)
The main thread function.
- void **CreateCpuThread** ()
Creates a new CPU thread.
- void **ExitCpuThread** ()
Kills this thread.
- void **ContExe** ()
Executes needed amount of instructions.
- BOOL WINAPI **DllMain** (HANDLE hModule, DWORD ul_reason_for_call, LPVOID lpReserved)
Dll entry point.
- DLLEXPORT void __cdecl **InitDll** (INITDLLSTRUCT *ids, INITDLLINFO *idi)
Init's dll data structures.
- DLLEXPORT void __cdecl **OnMessage** (MESSAGEDLLSTRUCT *mds)
Receives registered messages from emu6502 application.
- DLLEXPORT void __cdecl **IOFunc** (UINT adr, unsigned char *data, iomode mode)
IO function mapped into some address in memory.

Variables

- UINT [Freq](#) = 1790000
- UINT [Frms](#) = 1790
- UINT [start](#)
- UINT [total](#)
- UINT [secstart](#)
- UINT [ticks](#)
- BOOL volatile [Running](#)
- THRSTATE volatile [ThreadState](#)
- INTR volatile [Interrupt](#)
- HANDLE [CpuThreadHandle](#)
- DWORD [CpuThreadID](#)
- HANDLE [hThisModule](#)
- MSGLIST [Msgs](#)
- INITDLLSTRUCT [info](#)

5.3.2 Function Documentation

5.3.2.1 void ContExe ()

Executes needed amount of instructions.

Executes appropriate amount of instructions according to given frequency. [secstart](#) must be set by [CreateCpuThread\(\)](#) when called the first time

See also:

[CreateCpuThread\(\)](#)

Definition at line 219 of file cpudll.c.

5.3.2.2 DWORD WINAPI CpuThreadFunc (LPVOID lpParam)

The main thread function.

This function is called by [CreateCpuThread\(\)](#)

Parameters:

lpParam Value of this param doesn't matter

Returns :

Nothing

Definition at line 202 of file cpudll.c.

5.3.2.3 void CreateCpuThread ()

Creates a new CPU thread.

Creates a new thread in which processor is running. Called by [DllMain](#) when loading this dll.

See also:

[CpuThreadFunc](#) , [DllMain](#) , [ExitCpuThread](#)

Definition at line 187 of file cpudll.c.

5.3.2.4 **BOOL APIENTRY DllMain (HANDLE *hModule*, DWORD *ul_reason_for_call*, LPVOID *lpReserved*)**

Dll entry point.

Definition at line 62 of file cpudll.c.

5.3.2.5 **void ExitCpuThread ()**

Kills this thread.

See also:

[CreateCpuThread\(\)](#)

Definition at line 250 of file cpudll.c.

5.3.2.6 **DLLEXPORT void __cdecl InitDll (INITDLLSTRUCT * *ids*, INTDLLINFO * *idf*)**

Inits dll data structures.

This function is called after dll is loaded. You have to implement this function. Useful informations are passed in its parameter. Also create, fill with messages to be caught and return [MSGLIST](#) structure.

Parameters:

ids An [INITDLLSTRUCT](#) structure containig informations about calling emu6502 application.

idf An [INTDLLINFO](#) structure which should be filled in this function

Note: if io memory boundaries to be reserved for this plugin are in conflict with some other plugin, previous settings will be overwritten!

Returns :

Filled [MSGLIST](#) of messages to be caught and forwarded to this plugin

See also:

[OnMessage\(MESSAGEDLLSTRUCT* mds\)](#) , [INITDLLSTRUCT](#) , [INTDLLINFO](#)

Definition at line 75 of file cpudll.c.

5.3.2.7 **DLLEXPORT void __cdecl IOFunc (UINT *adr*, unsigned char * *data*, **iomode** *mode*)**

IO function mapped into some adress in memory.

Called directly when accesing adress to which is this function bound. Always write this function.

Parameters:

adr Adress being accessed

data Byte being read/written

mode see [iomode](#)

Definition at line 172 of file cpudll.c.

5.3.2.8 DLLEXPORT void __cdecl OnMessage (MESSAGEDLLSTRUCT * mds)

Receives registered messages from emu6502 application.

Called from another thread!!

Definition at line 113 of file cpudll.c.

5.3.3 Variable Documentation

5.3.3.1 HANDLE CpuThreadHandle

Handle of this thread

Definition at line 52 of file cpudll.c.

5.3.3.2 DWORD CpuThreadID

ID of this thread

Definition at line 53 of file cpudll.c.

5.3.3.3 UINT Freq = 1790000

Frequency of processor

Definition at line 24 of file cpudll.c.

5.3.3.4 UINT Frms = 1790

No. of instructions per milisecond

Definition at line 26 of file cpudll.c.

5.3.3.5 INTR volatile Interrupt

TRUE if interrupt occurs

Definition at line 50 of file cpudll.c.

5.3.3.6 BOOL volatile Running

Can be used by two threads!

Definition at line 45 of file cpudll.c.

5.3.3.7 UINT secstart

Helper variable used for determining and correcting the CPU speed

Definition at line 40 of file cpudll.c.

5.3.3.8 UINT start

Start time of CPUs execution

Definition at line 38 of file cpudll.c.

5.3.3.9 THRSTATE volatile ThreadState

Used for synchronization (to make sure, thread has correctly ended)

Definition at line 46 of file cpudll.c.

5.3.3.10 UINT ticks

Last value of GetTickCount();

Definition at line 41 of file cpudll.c.

5.3.3.11 UINT total

No of ticks from @start

Definition at line 39 of file cpudll.c.

5.4 cpuuidll.c File Reference

5.4.1 Detailed Description

Menues and toolbar buttons for controlling cpu

Definition in file `cpuuidll.c`:
`#include " ../dllkit/dllkit.h"`

`#include " ../usrmsgs.h"`

`#include "resource.h"`

`#include <commctrl.h>`

`#include <stdio.h>`

`#include <tchar.h>`

Functions

- void `SetStatusRunning ()`
enables/disables accordant toolbar buttons and menu entries
- void `SetStatusStopped ()`
enables/disables accordant toolbar buttons and menu entries
- BOOL WINAPI `DllMain (HANDLE hModule, DWORD ul_reason_for_call, LPVOID lpReserved)`
Entry point of this dll.
- DLLEXPORT void __cdecl `InitDll (INITDLLSTRUCT *ids, INITDLLINFO *idi)`
Inits dll data structures.
- DLLEXPORT void __cdecl `OnMessage (MESSAGEDLLSTRUCT *mds)`
Receives registered messages from emu6502 application.
- DLLEXPORT void __cdecl `IOFunc (UINT adr, unsigned char *data, iomode mode)`
IO function mapped into some adress in memory.

Variables

- HANDLE `hThisModule`
Handle to instance of this module.
- MSGLIST `Msgs`
Contains list of messages to be forwarded to this plugin.
- INITDLLSTRUCT `info`
Local copy of a structure passed to this plugin containging various info on the main emu6502 window and application.
- HMENU `hMenu`

Handle to the main menu bar.

- HMENU [hCpuMenu](#)

Handle to the CPU menu.

- BOOL **Running**

5.4.2 Function Documentation

5.4.2.1 BOOL APIENTRY DllMain (HANDLE *hModule*, DWORD *ul_reason_for_call*, LPVOID *lpReserved*)

Entry point of this dll.

Definition at line 36 of file cpuuidll.c.

5.4.2.2 DLLEXPORT void __cdecl InitDll (INITDLLSTRUCT * *ids*, INTDLLINFO * *idf*)

Inits dll data structures.

This function is called after dll is loaded. You have to implement this function. Useful informations are passed in its parameter. Also create, fill with messages to be caught and return [MSGLIST](#) structure.

Parameters:

ids An [INITDLLSTRUCT](#) structure containig informations about calling emu6502 application.

idf An [INTDLLINFO](#) structure which should be filled in this function

Note: if io memory boundaries to be reserved for this plugin are in conflict with some other plugin, previous settings will be overwritten!

Returns :

Filled [MSGLIST](#) of messages to be caught and forwarded to this plugin

See also:

[OnMessage\(MESSAGEDLLSTRUCT* mds\)](#) , [INITDLLSTRUCT](#) , [INTDLLINFO](#)

Definition at line 51 of file cpuuidll.c.

5.4.2.3 DLLEXPORT void __cdecl IOFunc (UINT *adr*, unsigned char * *data*, [iomode](#) *mode*)

IO function mapped into some adress in memory.

Called directly when accesing adress to which is this function bound. Always write this function.

Parameters:

adr Adress being accessed

data Byte being read/written

mode see [iomode](#)

Definition at line 214 of file cpuuidll.c.

5.4.2.4 DLLEXPORT void __cdecl OnMessage ([MESSAGEDLLSTRUCT](#) * *mds*)

Receives registered messages from emu6502 application.

This function is called by emu6502 when some message event registered by *InitDll* occurs. Implement this function even when you don't want to catch any messages.

Parameters:

mds A [MESSAGEDLLSTRUCT](#) structure containing informations about incoming message.

See also:

[InitDll](#) , [MESSAGEDLLSTRUCT](#)

Definition at line 126 of file `cpuuidll.c`.

5.4.2.5 void SetStatusRunning ()

enables/disables accordant toolbar buttons and menu entries

Definition at line 174 of file `cpuuidll.c`.

5.4.2.6 void SetStatusStopped ()

enables/disables accordant toolbar buttons and menu entries

Definition at line 194 of file `cpuuidll.c`.

5.4.3 Variable Documentation

5.4.3.1 HMENU hCpuMenu

Handle to the CPU menu.

Definition at line 24 of file `cpuuidll.c`.

5.4.3.2 HMENU hMenu

Handle to the main menu bar.

Definition at line 22 of file `cpuuidll.c`.

5.4.3.3 HANDLE hThisModule

Handle to instance of this module.

Definition at line 14 of file `cpuuidll.c`.

5.4.3.4 [INITDLLSTRUCT](#) info

Local copy of a structure passed to this plugin containing various info on the main emu6502 window and application.

Definition at line 19 of file `cpuuidll.c`.

5.4.3.5 **MSGLIST** Msgs

Contains list of messages to be forwarded to this plugin.

Definition at line 16 of file cpuidll.c.

5.5 debugdll.c File Reference

5.5.1 Detailed Description

Simple debugger & disassembler for emu6502

Definition in file [debugdll.c](#).#include "../dllkit/dllkit.h"

```
#include "../usrmsgs.h"
```

```
#include "resource.h"
```

```
#include "instructions.h"
```

```
#include <windows.h>
```

```
#include <commctrl.h>
```

```
#include <stdlib.h>
```

```
#include <stdio.h>
```

Data Structures

- struct [Regs](#)
Registers.

Defines

- #define **ADR**(X, Y) (((Y) << 8) + (X))
- #define **FN** 0x80
- #define **FV** 0x40
- #define **FB** 0x10
- #define **FD** 0x08
- #define **FI** 0x04
- #define **FZ** 0x02
- #define **FC** 0x01
- #define **GETFLAG**(FL) ((regs → P & (FL)) == 0) ? 0 : 1

Functions

- void [StartDebug](#) ()
Starts debug.
- void [DisAsm](#) (int lower, int upper, HANDLE fil, int act)
Disassembler.
- void [DumpMemHex](#) (HANDLE fil, UINT adr1, UINT adr2)
Printer of memory in hex format.
- void [DumpMem](#) (HANDLE fil, UINT adr1, UINT adr2)
Printer of memory in raw format.

- void [printhelp](#) ()
Displays usage instructions.
- HANDLE [ofile](#) ()
Shows save dialog.
- HANDLE [orfile](#) ()
Shows open dialog.
- void [SetReg](#) (Regs *regs, char *buf)
Helper procedure for StartDebug.
- void [printregs](#) (Regs *regs)
Prints actual state of registers onto console.
- void [printfil](#) (HANDLE fil, char *str)
Prints message onto screen or into specified stream.
- void [rinterval](#) (UINT *adr1, UINT *adr2)
Reads an address interval and controls if it is correct.
- UINT [rhex](#) (char *msg)
Reads an hexadecimal integer and controls if it is correct.
- void [mklpstr](#) (char *str)
Converts strings read from console to null terminated strings.
- void [print](#) (char *str)
Prints given message onto the console.
- void [rline](#) (char *str)
Reads one command from console.
- void [printact](#) (Regs *regs)
Prints registers and disassembled position in memory.
- BOOL WINAPI [DllMain](#) (HANDLE hModule, DWORD ul_reason_for_call, LPVOID lpReserved)
Entry point of this dll.
- DLLEXPORT void __cdecl [InitDll](#) (INITDLLSTRUCT *ids, INITDLLINFO *idi)
Inits dll data structures.
- DLLEXPORT void __cdecl [OnMessage](#) (MESSAGEDLLSTRUCT *mds)
Receives registered messages from emu6502 application.
- DLLEXPORT void __cdecl [IOFunc](#) (UINT adr, unsigned char *data, [iomode](#) mode)
IO function mapped into some address in memory.

Variables

- HANDLE **hThisModule**
- **MSGLIST** **Msgs**
- **INITDLLSTRUCT** **info**
- HMENU **hMenu**

Handle to the main menu bar.

- HMENU **hDebugMenu**

*Handle to local debug menu which is appended to **hMenu**.*

5.5.2 Define Documentation

5.5.2.1 #define FB 0x10

BRK flag

Definition at line 19 of file debugdll.c.

5.5.2.2 #define FC 0x01

Carry flag

Definition at line 23 of file debugdll.c.

5.5.2.3 #define FD 0x08

Decimal flag

Definition at line 20 of file debugdll.c.

5.5.2.4 #define FI 0x04

Interrupt flag

Definition at line 21 of file debugdll.c.

5.5.2.5 #define FN 0x80

Negative flag

Definition at line 17 of file debugdll.c.

5.5.2.6 #define FV 0x40

Overflow flag

Definition at line 18 of file debugdll.c.

5.5.2.7 #define FZ 0x02

Zero flag

Definition at line 22 of file debugdll.c.

5.5.3 Function Documentation

5.5.3.1 void DisAsm (int *lower*, int *upper*, HANDLE *fil*, int *act*)

Disassembler.

Disassembles given range of memory and writes it into the stream specified

Parameters:

lower, *upper* Range of memory to be disassembled

fil Handle to stream to write disassembled data into

act Adress of instruction to be highlighted by adding >>> before it

Definition at line 510 of file debugdll.c.

5.5.3.2 BOOL APIENTRY DllMain (HANDLE *hModule*, DWORD *ul_reason_for_call*, LPVOID *lpReserved*)

Entry point of this dll.

Definition at line 149 of file debugdll.c.

5.5.3.3 void DumpMem (HANDLE *fil*, UINT *adr1*, UINT *adr2*)

Printer of memory in raw format.

Prints given range of memory into given stream in raw format

Parameters:

adr1, *adr2* Range of memory to be printed

fil Stream to write the memory into

Definition at line 639 of file debugdll.c.

5.5.3.4 void DumpMemHex (HANDLE *fil*, UINT *adr1*, UINT *adr2*)

Printer of memory in hex format.

Prints given range of memory into given stream in hexadecimal format

Parameters:

adr1, *adr2* Range of memory to be printed

fil Stream to write the memory into

Definition at line 646 of file debugdll.c.

5.5.3.5 DLLEXPORT void __cdecl InitDll (INITDLLSTRUCT * *ids*, INTDLLINFO * *idf*)

Inits dll data structures.

This function is called after dll is loaded. You have to implement this function. Useful informations are passed in its parameter. Also create, fill with messages to be caught and return *MSGLIST* structure.

Parameters:

ids An *INITDLLSTRUCT* structure containig informations about calling emu6502 application.

idf An *INTDLLINFO* structure which should be filled in this function

Note: if io memory boundaries to be reserved for this plugin are in conflict with some other plugin, previous settings will be overwritten!

Returns :

Filled *MSGLIST* of messages to be caught and forwarded to this plugin

See also:

[OnMessage\(MESSAGEDLLSTRUCT* mds\)](#) , [INITDLLSTRUCT](#) , [INTDLLINFO](#)

Definition at line 164 of file debugdll.c.

5.5.3.6 DLLEXPORT void __cdecl IOFunc (UINT *adr*, unsigned char * *data*, *iomode mode*)

IO function mapped into some adress in memory.

Called directly when accesing adress to which is this function bound. Always write this function.

Parameters:

adr Adress being accessed

data Byte being read/written

mode see *iomode*

Definition at line 238 of file debugdll.c.

5.5.3.7 void mklpstr (char * *str*)

Converts strings read from console to null terminated strings.

Definition at line 775 of file debugdll.c.

5.5.3.8 HANDLE ofile ()

Shows save dialog.

Shows save dialog and opens selected file.

Returns :

Handle to just opened stream of the selected file

Definition at line 693 of file debugdll.c.

5.5.3.9 DLLEXPORT void __cdecl OnMessage (MESSAGEDLLSTRUCT * mds)

Receives registered messages from emu6502 application.

This function is called by emu6502 when some message event registered by *InitDll* occurs. Implement this function even when you don't want to catch any messages.

Parameters:

mds A *MESSAGEDLLSTRUCT* structure containing information about incoming message.

See also:

[InitDll](#) , [MESSAGEDLLSTRUCT](#)

Definition at line 210 of file debugdll.c.

5.5.3.10 HANDLE orfile ()

Shows open dialog.

Shows open dialog and opens selected file.

Returns :

Handle to just opened stream of the selected file

Definition at line 660 of file debugdll.c.

5.5.3.11 void print (char * str)

Prints given message onto the console.

Definition at line 793 of file debugdll.c.

5.5.3.12 void printact (Regs * regs)

Prints registers and disassembled position in memory.

Definition at line 727 of file debugdll.c.

5.5.3.13 void printfil (HANDLE fil, char * str)

Prints message onto screen or into specified stream.

If the fil stream is NULL, the message will be printed onto console

Parameters:

fil Handle to a stream or NULL

str Message to be printed

Definition at line 782 of file debugdll.c.

5.5.3.14 void printhelp ()

Displays usage instructions.

Definition at line 614 of file debugdll.c.

5.5.3.15 void printregs ([Regs](#) * *regs*)

Prints actual state of registers onto console.

Definition at line 738 of file debugdll.c.

5.5.3.16 UINT rhex (char * *msg*)

Reads an hexadecimal integer and controls if it is correct.

Definition at line 757 of file debugdll.c.

5.5.3.17 void rinterval (UINT * *adr1*, UINT * *adr2*)

Reads an address interval and controls if it is correct.

Definition at line 749 of file debugdll.c.

5.5.3.18 void rline (char * *str*)

Reads one command from console.

Definition at line 799 of file debugdll.c.

5.5.3.19 void SetReg ([Regs](#) * *regs*, char * *buf*)

Helper procedure for *StartDebug*.

Parses registers modifying commands.

Parameters:

regs Pointer to the [Regs](#) structure used by the CPU plugin

buf Pointer to the command user has entered

Definition at line 402 of file debugdll.c.

5.5.3.20 void StartDebug ()

Starts debug.

Retrieves pointer to registers from the cpu plugin, if not present, shows an error message and stops debugging .

Definition at line 243 of file debugdll.c.

5.5.4 Variable Documentation

5.5.4.1 HMENU *hDebugMenu*

Handle to local debug menu which is appended to *hMenu*.

Definition at line 42 of file debugdll.c.

5.5.4.2 HMENU hMenu

Handle to the main menu bar.

Definition at line 40 of file debugdll.c.

5.6 dllkit.c File Reference

5.6.1 Detailed Description

Implements functions from *dllkit.h*. Library for creating emu6502 plugins.

Definition in file `dllkit.c`.`#include "dllkit.h"`

Functions

- **BOOL CreateMsgList (MSGLIST *MsgList, int MessageCount)**
Allocates MsgList.
- **void FreeMsgList (MSGLIST MsgList)**
Deallocates MsgList Calls free on MsgList.
- **void AddMessage (MSGLIST MsgList, UINT Message)**
Adds Message to the given MsgList.

5.6.2 Function Documentation

5.6.2.1 void AddMessage (MSGLIST MsgList, UINT Message)

Adds Message to the given MsgList.

Adds Message to the given MsgList. The message shouldn't be *ENDOFMSGLIST*.

Parameters:

MsgList List of messages to which the Message should be added.

Message Message which will be added to the MsgList

Definition at line 27 of file `dllkit.c`.

5.6.2.2 BOOL CreateMsgList (MSGLIST * MsgList, int MessageCount)

Allocates MsgList.

Allocates MsgList for holding MessageCount of messages. MsgList takes (MessageCount + 1) * sizeof(UINT) bytes and first item is set to *ENDOFMSGLIST*.

Parameters:

MsgList A MSGLIST variable to be allocated.

MessageCount Max. count of messages to be held within the list.

Returns :

TRUE if succesful, otherwise FALSE.

Definition at line 9 of file `dllkit.c`.

5.6.2.3 void FreeMsgList (MSGLIST *MsgList*)

Deallocates MsgList Calls free on MsgList.

Parameters:

MsgList A MSGLIST variable to be deallocated.

Definition at line 22 of file dllkit.c.

5.7 dllkit.h File Reference

5.7.1 Detailed Description

Header file of the 6502 dll development kit. Library for creating emu6502 plugins.

Definition in file `dllkit.h`.`#include <windows.h>`

Data Structures

- struct `INITDLLINFO`
Structure containing various informations returned by InitDll.
- struct `INITDLLSTRUCT`
Initial informations passed to plugin when initializing.
- struct `MESSAGEDLLSTRUCT`
Message structure passed to plugins, when registered message occurred.

Defines

- #define `DLLEXPORT` `__declspec(dllexport)`
- #define `IOL0` `0xD000`
Lower boundary of memory serving for device mapping.
- #define `IOHI` `0xD7FF`
Higher boundary of memory serving for device mapping.
- #define `NMIID` `0x020A`
Register in memory which serves to identify the source of last interrupt.
- #define `IRQID` `0x020B`
- #define `RESETID` `0x020C`
- #define `ISVBI` `0x020D`
- #define `ENDOFMSGLIST` `0xBFFF`
Marks the end of the list of messages `MSGLIST`.

Typedefs

- typedef `UINT * MSGLIST`
List of messages.

Enumerations

- enum `iomode` { `IOREAD`, `IOWRITE` }
Determines, if `iofunc` is called when reading or writing.

Functions

- BOOL [CreateMsgList](#) ([MSGLIST](#) *MsgList, int MessageCount)
Allocates MsgList.
- void [FreeMsgList](#) ([MSGLIST](#) MsgList)
Deallocates MsgList Calls free on MsgList.
- void [AddMessage](#) ([MSGLIST](#) MsgList, UINT Message)
Adds Message to the given MsgList.
- DLLEXPORT void __cdecl [IOFunc](#) (UINT adr, unsigned char *data, [iomode](#) mode)
IO function mapped into some adress in memory.
- DLLEXPORT void __cdecl [InitDll](#) ([INITDLLSTRUCT](#) *ids, [INITDLLINFO](#) *idf)
Inits dll data structures.
- DLLEXPORT void __cdecl [OnMessage](#) ([MESSAGEDLLSTRUCT](#) *mds)
Receives registered messages from emu6502 application.

5.7.2 Define Documentation

5.7.2.1 #define ENDOFMSGLIST 0xBFFF

Marks the end of the list of messages [MSGLIST](#).

Definition at line 36 of file dllkit.h.

5.7.2.2 #define IOHI 0xD7FF

Higher boundary of memory serving for device mapping.

Definition at line 18 of file dllkit.h.

5.7.2.3 #define IOLO 0xD000

Lower boundary of memory serving for device mapping.

Definition at line 16 of file dllkit.h.

5.7.2.4 #define NMIID 0x020A

Register in memory which serves to identify the source of last interrupt.

Definition at line 21 of file dllkit.h.

5.7.3 Typedef Documentation

5.7.3.1 typedef UINT* MSGLIST

List of messages.

Used for specifying messages to be caught or to be sent. Length of this list should be the number of messages to be held within + 1, and the list should end with *ENDOFMSGLIST*

Definition at line 33 of file dllkit.h.

5.7.4 Enumeration Type Documentation

5.7.4.1 enum iomode

Determines, if iofunc is called when reading or writing.

Definition at line 115 of file dllkit.h.

5.7.5 Function Documentation

5.7.5.1 void AddMessage (**MSGLIST** *MsgList*, **UINT** *Message*)

Adds Message to the given MsgList.

Adds Message to the given MsgList. The message shouldn't be *ENDOFMSGLIST*.

Parameters:

MsgList List of messages to which the Message should be added.

Message Message which will be added to the MsgList

Definition at line 27 of file dllkit.c.

5.7.5.2 **BOOL** CreateMsgList (**MSGLIST** * *MsgList*, **int** *MessageCount*)

Allocates MsgList.

Allocates MsgList for holding MessageCount of messages. MsgList takes (MessageCount + 1) * sizeof(UINT) bytes and first item is set to *ENDOFMSGLIST*.

Parameters:

MsgList A MSGLIST variable to be allocated.

MessageCount Max. count of messages to be held within the list.

Returns :

TRUE if succesful, otherwise FALSE.

Definition at line 9 of file dllkit.c.

5.7.5.3 void FreeMsgList (**MSGLIST** *MsgList*)

Deallocates MsgList Calls free on MsgList.

Parameters:

MsgList A MSGLIST variable to be deallocated.

Definition at line 22 of file dllkit.c.

5.7.5.4 DLLEXPORT void __cdecl InitDll (INITDLLSTRUCT * *ids*, INITDLLINFO * *idi*)

Inits dll data structures.

Todo:

free VideoMsgs

Definition at line 41 of file genericdll.c.

5.7.5.5 DLLEXPORT void __cdecl IOFunc (UINT *adr*, unsigned char * *data*, *iomode mode*)

IO function mapped into some adress in memory.

Called directly when accesing adress to which is this function bound. Always write this function.

Parameters:

adr Adress being accessed

data Byte being read/written

mode see *iomode*

Definition at line 95 of file genericdll.c.

5.7.5.6 DLLEXPORT void __cdecl OnMessage (MESSAGEDLLSTRUCT * *mds*)

Recieves registered messages from emu6502 application.

This function is called by emu6502 when some message event registered by *InitDll* occurs. Implement this function even when you don't want to catch any messages.

Parameters:

mds A *MESSAGEDLLSTRUCT* structure containing informations about incoming message.

See also:

[InitDll](#) , [MESSAGEDLLSTRUCT](#)

Definition at line 69 of file genericdll.c.

5.8 dlluser.h File Reference

5.8.1 Detailed Description

Include this file if you want to use emu6502 plugin dlls.

Definition in file `dlluser.h`.`#include <windows.h>`

Data Structures

- struct `INITDLLINFO`
Structure containing various informations returned by InitDll.
- struct `INITDLLSTRUCT`
Initial informations passed to plugin when initializing.
- struct `MESSAGEDLLSTRUCT`
Message structure passed to plugins, when registered message occurred.

Defines

- `#define ENDOFMSGLIST 0xBFFF`
Marks the end of the list of messages `MSGLIST`.

Typedefs

- typedef `UINT * MSGLIST`
List of messages.
- typedef `void(* pIOFunc)(UINT adr, unsigned char *data, iomode mode)`
IO function mapped into some adress in memory.
- typedef `void(* pInitDllProc)(INITDLLSTRUCT *, INITDLLINFO *)`
emu6502 plugin dll initialization routine pointer
- typedef `void(* pForwardDllMsgProc)(MESSAGEDLLSTRUCT *)`
emu6502 plugin dll message handling routine pointer

Enumerations

- enum `iomode { IOREAD, IOWRITE }`
Determines, if iofunc is called when reading or writing.

5.8.2 Define Documentation

5.8.2.1 #define ENDOFMSGLIST 0xBFFF

Marks the end of the list of messages *MSGLIST*.

Definition at line 20 of file dlluser.h.

5.8.3 Typedef Documentation

5.8.3.1 typedef UINT* MSGLIST

List of messages.

Generic list of messages, used for specifying messages to be caught or to be sent. Length of this list should be the number of messages to be held within + 1, and the list should end with *ENDOFMSGLIST*

Definition at line 17 of file dlluser.h.

5.8.3.2 typedef void(* pForwardDllMsgProc)(MESSAGE DLLSTRUCT*)

emu6502 plugin dll message handling routine pointer

Definition at line 89 of file dlluser.h.

5.8.3.3 typedef void(* pInitDllProc)(INITDLLSTRUCT*, INITDLLINFO*)

emu6502 plugin dll initialization routine pointer

Definition at line 86 of file dlluser.h.

5.8.3.4 typedef void(* pIOFunc)(UINT adr, unsigned char* data, iomode mode)

IO function mapped into some address in memory.

Called directly when accessing address to which is this function bound

Parameters:

adr Address being accessed

data Byte being read/written

mode see *iomode*

See also:

iomode , *memio*

Definition at line 83 of file dlluser.h.

5.8.4 Enumeration Type Documentation

5.8.4.1 enum iomode

Determines, if *iofunc* is called when reading or writing.

Definition at line 71 of file dlluser.h.

5.9 emu6502.c File Reference

5.9.1 Detailed Description

Main framework of the emu6502 applictaion, contains window and plugin management.

Definition in file [emu6502.c](#).#include <windows.h>

```
#include <stdlib.h>
#include <malloc.h>
#include <memory.h>
#include <tchar.h>
#include <commdlg.h>
#include <commctrl.h>
#include <stdio.h>
#include "emu6502.h"
```

Defines

- #define **WIN32_LEAN_AND_MEAN**
- #define **MAX_LOADSTRING** 512
- #define **CR** 13
- #define **LF** 10

Functions

- int APIENTRY [WinMain](#) (HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)
Entry point of this app.
- ATOM [MyRegisterClass](#) (HINSTANCE hInstance)
Registers the main window class.
- BOOL [InitInstance](#) (HINSTANCE hInstance, int nCmdShow)
Inits & shows the main window.
- HWND [CreateToolBar](#) (HWND hParentWnd)
Creates the main toolbar.
- LRESULT CALLBACK [WndProc](#) (HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)
Main message handling routine.
- LRESULT CALLBACK [About](#) (HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam)
About dialog's DialogProc.
- BOOL [LoadMemFile](#) (HWND hWnd)

Loads memory image.

- void **error** (HWND hWnd, UINT uID, TCHAR *name)
Error reporting proc.
- BOOL **ForwardDllMessage** (UINT message, WPARAM wParam, LPARAM lParam, **DllMsgList** *actlist)
Forwards message to registered plugin OnMessage functions.
- void **LoadPlugins** (**DllMsgList** **list, **PtrList** **InstList)
Loads plugins named in plugins.txt.
- BOOL **LoadPlugin** (TCHAR *name, **DllMsgList** **list, **PtrList** **InstList)
Loads plugin of given name.
- void **AddDllMsg** (**DllMsgList** **list, UINT message, **pForwardDllMsgProc** FDMProc)
Adds message catching dll function to a dll message list.
- void **UnloadPlugins** (**PtrList** *list)
- void **AddToPtrList** (void *ptr, **PtrList** **list)
Allocates and adds new member to existing or NULL list.
- TCHAR * **GetNextStrToken** (TCHAR **buf)
Helper function for LoadPlugins.

Variables

- HINSTANCE **hInst**
Handle to instance of this app.
- HWND **hMainWnd**
Handle to the main window.
- HWND **hToolBar**
Handle to the toolbar.
- **DllMsgList** * **DMList** = NULL
List of messages and theirs catching dll functions.
- **PtrList** * **DHInstList** = NULL
List of dll handles.
- TCHAR **szTitle** [MAX_LOADSTRING]
- TCHAR **szWindowClass** [MAX_LOADSTRING]

5.9.2 Function Documentation

5.9.2.1 LRESULT CALLBACK About (HWND, UINT, WPARAM, LPARAM)

About dialog's DialogProc.

Definition at line 198 of file emu6502.c.

5.9.2.2 void AddDllMsg (DllMsgList ** list, UINT message, pForwardDllMsgProc FDMProc)

Adds message catching dll function to a dll message list.

Parameters:

list pointer to a *DllMsgList* list

message ID of message to be caught

FDMProc Function, to which should be the message forwarded Mostly function from plugin-dll.

See also:

ForwardDllMsgProc , [DllMsgList](#)

Definition at line 399 of file emu6502.c.

5.9.2.3 void AddToPtrList (void * node, PtrList ** list)

Allocates and adds new member to existing or NULL list.

Definition at line 446 of file emu6502.c.

5.9.2.4 HWND CreateToolBar (HWND hParentWnd)

Creates the main toolbar.

Creates main toolbar and adds one Open button to it

Parameters:

hParentWnd Handle to parent window

Returns :

Handle to the toolbar

Definition at line 127 of file emu6502.c.

5.9.2.5 void error (HWND hWnd, UINT uID, TCHAR * name)

Error reporting proc.

Shows a messagebox with given error message (from resources). This message can also contain s in this case the parameter name becomes effective and prints on the position of s.

Parameters:

hWnd Handle of the parent window

uID Resource ID of the error message
name Informations to be printed on the s position

Definition at line 268 of file emu6502.c.

5.9.2.6 **BOOL ForwardDllMessage** (UINT *message*, WPARAM *wParam*, LPARAM *lParam*, [DllMsgList](#) * *list*)

Forwards message to registered plugin OnMessage functions.

Parameters:

message ID of forwarded message
wParam wParam parameter of this message
lParam lParam parameter of this message
list List of messages and its catching functions. See [DllMsgList](#)

Returns :

TRUE if relevant function found, FALSE otherwise

Definition at line 277 of file emu6502.c.

5.9.2.7 **TCHAR*** **GetNextStrToken** (TCHAR ** *buf*)

Helper function for LoadPlugins.

Sideeffect - makes buf to point to the next item

Parameters:

buf TCHAR buffer containing list of items separated by CRLF and ended with \0.

Returns :

first item of the list or NULL if there isn't any more

See also:

[LoadPlugins](#)

Definition at line 454 of file emu6502.c.

5.9.2.8 **BOOL InitInstance** (HINSTANCE, int)

Inits & shows the main window.

Creates the main window, sizes it to fit the 256x256 canvas, creates toolbar by calling @CreateToolbar, and then loads plugins by calling *LoadPlugins*.

See also:

CreateToolbar, [LoadPlugins](#)

Definition at line 91 of file emu6502.c.

5.9.2.9 **BOOL LoadMemFile (HWND *hWnd*)**

Loads memory image.

Stops execution of the CPU, loads memory image, also controls it's length

Definition at line 216 of file emu6502.c.

5.9.2.10 **BOOL LoadPlugin (TCHAR * *name*, DllMsgList ** *list*, PtrList ** *InstList*)**

Loads plugin of given name.

Executes his InitDll functions, receives list of messages to be caught in this plugin and attaches it to the [DllMsgList](#) list.

Parameters:

name Name of plugin dll to be loaded.

list Pointer to a [DllMsgList](#) list in which will be message IDs and theirs catching procedures stored.

InstList List of handles to plugin-dlls loaded

See also:

[LoadPlugins](#), [ForwardDllMessage](#)

Definition at line 334 of file emu6502.c.

5.9.2.11 **void LoadPlugins (DllMsgList ** *list*, PtrList ** *InstList*)**

Loads plugins named in plugins.txt.

Goes through the plugins.txt file and loads named plugins by calling *LoadPlugin* procedure.

Parameters:

list Pointer to a [DllMsgList](#) list in which will be message IDs and theirs catching procedures stored.

InstList List of handles to plugin-dlls loaded

See also:

[LoadPlugin](#), [ForwardDllMessage](#)

Definition at line 303 of file emu6502.c.

5.9.2.12 **ATOM MyRegisterClass (HINSTANCE *hInstance*)**

Registers the main window class.

Definition at line 69 of file emu6502.c.

5.9.2.13 **int APIENTRY WinMain (HINSTANCE *hInstance*, HINSTANCE *hPrevInstance*, LPSTR *lpCmdLine*, int *nCmdShow*)**

Entry point of this app.

Definition at line 41 of file emu6502.c.

5.9.2.14 LRESULT CALLBACK WndProc (HWND *hWnd*, UINT *message*, WPARAM *wParam*, LPARAM *lParam*)

Main message handling routine.

Todo:

add TBSTYLE_TOOLTIPS

Definition at line 155 of file emu6502.c.

5.9.3 Variable Documentation

5.9.3.1 **PtrList*** DHInstList = NULL

List of dll handles.

Definition at line 34 of file emu6502.c.

5.9.3.2 **DIIMsgList*** DMList = NULL

List of messages and theirs catching dll functions.

Definition at line 32 of file emu6502.c.

5.9.3.3 **HINSTANCE** hInst

Handle to instance of this app.

Definition at line 25 of file emu6502.c.

5.9.3.4 **HWND** hMainWnd

Handle to the main window.

Definition at line 27 of file emu6502.c.

5.9.3.5 **HWND** hToolBar

Handle to the toolbar.

Definition at line 29 of file emu6502.c.

5.10 emu6502.h File Reference

5.10.1 Detailed Description

Header file for *emu6502.c*, contains only local forward and type declarations, not for reuse.

Definition in file *emu6502.h*.`#include "resource.h"`

```
#include "mem.h"
```

```
#include "dlluser.h"
```

```
#include "usrmsgs.h"
```

Data Structures

- struct **_DllMsgList**
- struct **_PtrList**

Typedefs

- typedef **_PtrList** **PtrList**
Generic list of pointers.
- typedef **_DllMsgList** **DllMsgList**
List of messages and theirs catching dll functions.

Functions

- int APIENTRY **WinMain** (HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)
Entry point of this app.
- ATOM **MyRegisterClass** (HINSTANCE hInstance)
Registers the main window class.
- BOOL **InitInstance** (HINSTANCE, int)
Inits & shows the main window.
- HWND **CreateToolBar** (HWND hParentWnd)
Creates the main toolbar.
- LRESULT CALLBACK **WndProc** (HWND, UINT, WPARAM, LPARAM)
Main message handling routine.
- LRESULT CALLBACK **About** (HWND, UINT, WPARAM, LPARAM)
About dialog's DialogProc.
- BOOL **LoadMemFile** (HWND hWnd)
Loads memory image.

- void **error** (HWND hWnd, UINT uID, TCHAR *name)
Error reporting proc.
- void **LoadPlugins** (DllMsgList **list, PtrList **InstList)
Loads plugins named in plugins.txt.
- BOOL **LoadPlugin** (TCHAR *name, DllMsgList **list, PtrList **InstList)
Loads plugin of given name.
- void **UnloadPlugins** (PtrList *list)
- BOOL **ForwardDllMessage** (UINT message, WPARAM wParam, LPARAM lParam, DllMsgList *list)
Forwards message to registered plugin OnMessage functions.
- void **AddDllMsg** (DllMsgList **list, UINT message, pForwardDllMsgProc FDMProc)
Adds message catching dll function to a dll message list.
- void **AddToPtrList** (void *node, PtrList **list)
Allocates and adds new member to existing or NULL list.
- TCHAR * **GetNextStrToken** (TCHAR **buf)
Helper function for LoadPlugins.

5.10.2 Typedef Documentation

5.10.2.1 typedef struct _DllMsgList DllMsgList

List of messages and theirs catching dll functions.

See also:

DllFuncNode , ForwardDllMsgProc

Definition at line 34 of file emu6502.h.

5.10.2.2 typedef struct _PtrList PtrList

Generic list of pointers.

Definition at line 22 of file emu6502.h.

5.10.3 Function Documentation

5.10.3.1 LRESULT CALLBACK About (HWND, UINT, WPARAM, LPARAM)

About dialog's DialogProc.

Definition at line 198 of file emu6502.c.

5.10.3.2 void AddDllMsg (DllMsgList ** list, UINT message, pForwardDllMsgProc FDMProc)

Adds message catching dll function to a dll message list.

Parameters:

list pointer to a *DllMsgList* list

message ID of message to be caught

FDMProc Function, to which should be the message forwarded Mostly function from plugin-dll.

See also:

ForwardDllMsgProc , [DllMsgList](#)

Definition at line 399 of file emu6502.c.

5.10.3.3 void AddToPtrList (void * node, PtrList ** list)

Allocates and adds new member to existing or NULL list.

Definition at line 446 of file emu6502.c.

5.10.3.4 HWND CreateToolBar (HWND hParentWnd)

Creates the main toolbar.

Creates main toolbar and adds one Open button to it

Parameters:

hParentWnd Handle to parent window

Returns :

Handle to the toolbar

Definition at line 127 of file emu6502.c.

5.10.3.5 void error (HWND hWnd, UINT uID, TCHAR * name)

Error reporting proc.

Shows a messagebox with given error message (from resources). This message can also contain s in this case the parameter name becomes effective and prints on the position of s.

Parameters:

hWnd Handle of the parent window

uID Resource ID of the error message

name Informations to be printed on the s position

Definition at line 268 of file emu6502.c.

5.10.3.6 **BOOL ForwardDllMessage** (UINT *message*, WPARAM *wParam*, LPARAM *lParam*, [DllMsgList](#) * *list*)

Forwards message to registered plugin OnMessage functions.

Parameters:

message ID of forwarded message

wParam wParam parameter of this message

lParam lParam parameter of this message

list List of messages and its catching functions. See [DllMsgList](#)

Returns :

TRUE if relevant function found, FALSE otherwise

Definition at line 277 of file emu6502.c.

5.10.3.7 **TCHAR* GetNextStrToken** (TCHAR ** *buf*)

Helper function for LoadPlugins.

Sideeffect - makes buf to point to the next item

Parameters:

buf TCHAR buffer containing list of items separated by CRLF and ended with \0.

Returns :

first item of the list or NULL if there isn't any more

See also:

[LoadPlugins](#)

Definition at line 454 of file emu6502.c.

5.10.3.8 **BOOL InitInstance** (HINSTANCE, int)

Init & shows the main window.

Creates the main window, sizes it to fit the 256x256 canvas, creates toolbar by calling @CreateToolbar, and then loads plugins by calling *LoadPlugins*.

See also:

CreateToolbar, [LoadPlugins](#)

Definition at line 91 of file emu6502.c.

5.10.3.9 **BOOL LoadMemFile** (HWND *hWnd*)

Loads memory image.

Stops execution of the CPU, loads memory image, also controls it's length

Definition at line 216 of file emu6502.c.

5.10.3.10 **BOOL LoadPlugin (TCHAR * name, DllMsgList ** list, PtrList ** InstList)**

Loads plugin of given name.

Executes his InitDll functions, receives list of messages to be caught in this plugin and attaches it to the *DllMsgList* list.

Parameters:

name Name of plugin dll to be loaded.

list Pointer to a *DllMsgList* list in which will be message IDs and theirs catching procedures stored.

InstList List of handles to plugin-dlls loaded

See also:

[LoadPlugins](#), [ForwardDllMessage](#)

Definition at line 334 of file emu6502.c.

5.10.3.11 **void LoadPlugins (DllMsgList ** list, PtrList ** InstList)**

Loads plugins named in plugins.txt.

Goes through the plugins.txt file and loads named plugins by calling *LoadPlugin* procedure.

Parameters:

list Pointer to a *DllMsgList* list in which will be message IDs and theirs catching procedures stored.

InstList List of handles to plugin-dlls loaded

See also:

[LoadPlugin](#), [ForwardDllMessage](#)

Definition at line 303 of file emu6502.c.

5.10.3.12 **ATOM MyRegisterClass (HINSTANCE hInstance)**

Registers the main window class.

Definition at line 69 of file emu6502.c.

5.10.3.13 **int APIENTRY WinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)**

Entry point of this app.

Definition at line 41 of file emu6502.c.

5.10.3.14 **LRESULT CALLBACK WndProc (HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)**

Main message handling routine.

Todo:

add TBSTYLE_TOOLTIPS

Definition at line 155 of file emu6502.c.

5.11 genericdll.c File Reference

5.11.1 Detailed Description

Template for creating plugins

Definition in file `genericdll.c`.`#include "../dllkit/dllkit.h"`

`#include "../usrmsgs.h"`

`#include <stdio.h>`

`#include <tchar.h>`

Defines

- `#define IOSETGET(GVAR, SVAR)`

Functions

- `BOOL APIENTRY DllMain` (`HANDLE hModule`, `DWORD ul_reason_for_call`, `LPVOID lpReserved`)
Entry point of this dll.
- `DLLEXPORT void __cdecl InitDll` (`INITDLLSTRUCT *ids`, `INITDLLINFO *idi`)
- `DLLEXPORT void __cdecl OnMessage` (`MESSAGEDLLSTRUCT *mds`)
- `DLLEXPORT void __cdecl IOFunc` (`UINT adr`, `unsigned char *data`, `iomode mode`)

Variables

- `HANDLE hThisModule`
Handle to instance of this module.
- `MSGLIST Msgs`
Contains list of messages to be forwarded to this plugin.
- `INITDLLSTRUCT info`
Local copy of a structure passed to this plugin containing various info on the main emu6502 window and application.

5.11.2 Define Documentation

5.11.2.1 `#define IOSETGET(GVAR, SVAR)`

Value:

```
if (mode == IOREAD) { \
    *data = GVAR; \
} \
else { \
    SVAR = *data; \
}
```

Maybe useful macro for IOFunc

Definition at line 83 of file genericdll.c.

5.11.3 Function Documentation

5.11.3.1 **BOOL** APIENTRY DllMain (**HANDLE** *hModule*, **DWORD** *ul_reason_for_call*, **LPCVOID** *lpReserved*)

Entry point of this dll.

Definition at line 20 of file genericdll.c.

5.11.3.2 **DLL_EXPORT void** __cdecl InitDll (**INITDLLSTRUCT** * *ids*, **INITDLLINFO** * *idi*)

See also:

[InitDll](#) in the [dllkit.h](#)

Definition at line 41 of file genericdll.c.

5.11.3.3 **DLL_EXPORT void** __cdecl IOFunc (**UINT** *adr*, **unsigned char** * *data*, **iomode** *mode*)

See also:

[InitDll](#) in the [dllkit.h](#)

Definition at line 95 of file genericdll.c.

5.11.3.4 **DLL_EXPORT void** __cdecl OnMessage (**MESSAGEDLLSTRUCT** * *mds*)

See also:

[OnMessage](#) in the [dllkit.h](#)

Definition at line 69 of file genericdll.c.

5.11.4 Variable Documentation

5.11.4.1 **HANDLE** *hThisModule*

Handle to instance of this module.

Definition at line 12 of file genericdll.c.

5.11.4.2 **INITDLLSTRUCT** *info*

Local copy of a structure passed to this plugin containing various info on the main emu6502 window and application.

Definition at line 17 of file genericdll.c.

5.11.4.3 **MSGLIST** Msgs

Contains list of messages to be forwarded to this plugin.

Definition at line 14 of file genericdll.c.

5.12 instructions.h File Reference

5.12.1 Detailed Description

Contains list of instructions and addressing modes

Definition in file [instructions.h](#).

Data Structures

- struct [_Instr](#)

Structure of one instruction entry in the [INSTRUCTIONS](#) list.

Typedefs

- typedef [_Instr Instr](#)

Structure of one instruction entry in the [INSTRUCTIONS](#) list.

Enumerations

- enum [AdrType](#) { [IMPL](#), [ACC](#), [IMM](#), [ABS](#), [ABSX](#), [ABSY](#), [ZP](#), [ZPX](#), [ZPY](#), [REL](#), [IND](#), [INDX](#), [INDY](#) }

Enum of addressing modes.

Variables

- const [Instr INSTRUCTIONS](#) [256]

List of instructions with theirs addressing modes and lengths.

5.12.2 Typedef Documentation

5.12.2.1 typedef struct [_Instr Instr](#)

Structure of one instruction entry in the [INSTRUCTIONS](#) list.

5.12.3 Enumeration Type Documentation

5.12.3.1 enum [AdrType](#)

Enum of addressing modes.

Definition at line 9 of file [instructions.h](#).

5.12.4 Variable Documentation

5.12.4.1 `const Instr INSTRUCTIONS[256]`

List of instructions with their addressing modes and lengths.

Definition at line 21 of file instructions.h.

5.13 mem.c File Reference

5.13.1 Detailed Description

Memory management is done in this file. It controls shared access to the memory structure via the CRITICAL_SECTION synchronization primitive. In trouble you can also uncomment the define DEBUG directive. It will create an out.txt file in which all accesses into memory will be written (except 0xA000 - 0xD000, but you can comment this)

Definition in file `mem.c`:

```
#include "mem.h"
```

```
#include <memory.h>
```

```
#include <windows.h>
```

```
#include <stdio.h>
```

Functions

- unsigned char `memgetb` (UINT adr)
memory reading function
- void `memsetb` (UINT adr, unsigned char bset)
memory writing function
- void `InitMem` ()
Zeroes `memio` and initializes `CSMemory`.
- void `SetIOFunc` (pIOFunc piof, UINT loadr, UINT hiadr)
Binds given io function to range of addresses.
- void `SetMemory` (char *buf)
Fills `memory` with given data.

Variables

- unsigned char `memory` [MEMSIZE+4]
Variable containing accessible memory +4 as pillow.
- pIOFunc `memio` [IOHI-IOLO+1]
IO part of memory.
- CRITICAL_SECTION `CSMemory`
A synchronization primitive used for shared acces to memory.

5.13.2 Function Documentation

5.13.2.1 void InitMem ()

Zeroes `memio` and initializes `CSMemory`.

Definition at line 97 of file mem.c.

5.13.2.2 unsigned char memgetb (UINT *adr*)

memory reading function

Calls iofunc when necessary

See also:

memsetb

Definition at line 32 of file mem.c.

5.13.2.3 void memsetb (UINT *adr*, unsigned char *byte*)

memory writing function

Calls iofunc when necessary

See also:

memgetb

Definition at line 73 of file mem.c.

5.13.2.4 void SetIOFunc (pIOFunc *piof*, UINT *loadr*, UINT *hiadr*)

Binds given io function to range of addresses.

Parameters:

piof An io function which should service reading and writing to these addresses

loadr Lower end of the range demanded

hiadr Higher end of the range demanded

Definition at line 109 of file mem.c.

5.13.2.5 void SetMemory (char * *buf*)

Fills *memory* with given data.

Definition at line 117 of file mem.c.

5.13.3 Variable Documentation

5.13.3.1 CRITICAL_SECTION CSMemory

A synchronization primitive used for shared acces to memory.

Definition at line 30 of file mem.c.

5.13.3.2 `pIOFunc` `memio[IOHI - IOLO + 1]`

IO part of memory.

Contains pointers to io functions of devices mapped into the io part of memory.

See also:

`iofunc`

Definition at line 28 of file `mem.c`.

5.13.3.3 `unsigned char` `memory[MEMSIZE + 4]`

Variable containing accessible memory +4 as pillow.

Definition at line 27 of file `mem.c`.

5.14 mem.h File Reference

5.14.1 Detailed Description

.h file to the [mem.c](#)

Definition in file [mem.h](#).#include "dlluser.h"

Defines

- #define **TRUE** 1
- #define **FALSE** 0
- #define **MEMSIZE** 65536
Size of memory.
- #define **PAGESIZE** 8
No. of bits of pagesize. Should correspond with MEMSIZE and PAGECOUNT.
- #define **PAGECOUNT** 256
No. of pages. Should correspond with MEMSIZE and PAGECOUNT.
- #define **IOLO** 0xD000
Lower boundary of memory serving for device mapping.
- #define **IOHI** 0xD7FF
Higher boundary of memory serving for device mapping.

Typedefs

- typedef unsigned int **UINT**

Functions

- unsigned char **memgetb** (UINT adr)
memory reading function
- void **memsetb** (UINT adr, unsigned char byte)
memory writing function
- void **SetMemory** (char *buf)
Fills memory with given data.
- void **InitMem** ()
Zeroes memio and initializes CSMemory.
- void **SetIOFunc** (pIOFunc piof, UINT loadr, UINT hiadr)
Binds given io function to range of addresses.

Variables

- unsigned char `memory` []
Variable containing accessible memory +4 as pillow.
- CRITICAL_SECTION `CSMemory`
A synchronization primitive used for shared acces to memory.
- `pIOFunc memio` [IOHI-IOLO+1]
IO part of memory.

5.14.2 Define Documentation

5.14.2.1 `#define IOHI 0xD7FF`

Higher boundary of memory serving for device maping.

Definition at line 26 of file mem.h.

5.14.2.2 `#define IOLO 0xD000`

Lower boundary of memory serving for device maping.

Definition at line 24 of file mem.h.

5.14.2.3 `#define MEMSIZE 65536`

Size of memory.

Definition at line 17 of file mem.h.

5.14.2.4 `#define PAGECOUNT 256`

No. of pages. Should correspond with *MEMSIZE* and *PAGECOUNT*.

Definition at line 21 of file mem.h.

5.14.2.5 `#define PAGESIZE 8`

No. of bits of pagesize. Should correspond with *MEMSIZE* and *PAGECOUNT*.

Definition at line 19 of file mem.h.

5.14.3 Function Documentation

5.14.3.1 `void InitMem ()`

Zeroes *memio* and initializes *CSMemory*.

Definition at line 97 of file mem.c.

5.14.3.2 unsigned char memgetb (UINT *adr*)

memory reading function

Calls iofunc when necessary

See also:

memsetb

Definition at line 32 of file mem.c.

5.14.3.3 void memsetb (UINT *adr*, unsigned char *byte*)

memory writing function

Calls iofunc when necessary

See also:

memgetb

Definition at line 73 of file mem.c.

5.14.3.4 void SetIOFunc (pIOFunc *piof*, UINT *loadr*, UINT *hiadr*)

Binds given io function to range of addresses.

Parameters:

piof An io function which should service reading and writing to these addresses

loadr Lower end of the range demanded

hiadr Higher end of the range demanded

Definition at line 109 of file mem.c.

5.14.3.5 void SetMemory (char * *buf*)

Fills *memory* with given data.

Definition at line 117 of file mem.c.

5.14.4 Variable Documentation

5.14.4.1 CRITICAL_SECTION CSMemory

A synchronization primitive used for shared acces to memory.

Definition at line 32 of file mem.h.

5.14.4.2 pIOFunc memio[IOHI - IOLO + 1]

IO part of memory.

Containins pointers to io functions of devices mapped into the io part of memory.

See also:

iofunc

Definition at line 41 of file mem.h.

5.14.4.3 unsigned char memory[]

Variable containing accessible memory +4 as pillow.

Definition at line 29 of file mem.h.

5.15 perifdll.c File Reference

5.15.1 Detailed Description

Handles events from misc periferies (at now keyboard only) and also contains random number generator

Definition in file `perifdll.c`:
`#include "../dllkit/dllkit.h"`

`#include "../usrmsgs.h"`

`#include <tchar.h>`

`#include <stdlib.h>`

Defines

- `#define KBSTAT IOLO + 21`
- `#define SHIFTSTAT IOLO + 22`
- `#define RANDOM IOLO + 23`
- `#define ADR(X, Y) (((X) << 8) + (Y))`
- `#define SHIFTFLAG 0x01`
- `#define CTRLFLAG 0x02`
- `#define IOSETGET(GVAR, SVAR)`

Functions

- `BOOL WINAPI DllMain (HANDLE hModule, DWORD ul_reason_for_call, LPVOID lpReserved)`
- `DLLEXPORT void __cdecl InitDll (INITDLLSTRUCT *ids, INITDLLINFO *idi)`
- `DLLEXPORT void __cdecl OnMessage (MESSAGEDLLSTRUCT *mds)`
- `DLLEXPORT void __cdecl IOFunc (UINT adr, unsigned char *data, iomode mode)`

Variables

- unsigned char `KbMap` [256]
Translation table.
- volatile unsigned char `Key` = 0
- volatile unsigned char `KbState` = 0
- `HANDLE hThisModule`
- `MSGLIST Msgs`
- `INITDLLSTRUCT info`

5.15.2 Define Documentation

5.15.2.1 `#define IOSETGET(GVAR, SVAR)`

Value:

```

if (mode == IOREAD) { \
    *data = GVAR; \
} \
else { \
    SVAR = *data; \
}

```

Definition at line 150 of file perfdll.c.

5.15.3 Function Documentation

5.15.3.1 DLLEXPORT void __cdecl InitDll ([INITDLLSTRUCT](#) * *ids*, [INITDLLINFO](#) * *idi*)

Todo:

free VideoMsgs

Definition at line 81 of file perfdll.c.

5.15.4 Variable Documentation

5.15.4.1 unsigned char KbMap[256]

Initial value:

```

{
    255, 255, 255, 255, 255, 255, 255, 255, 30, 125, 255,
    255, 255, 0, 255, 255, 255, 255, 255, 255, 255, 1,
    255, 255, 2, 3, 4, 5, 27, 255, 255, 255,
    255, 32, 255, 255, 255, 255, 255, 30, 28, 31, 29,
    255, 255, 255, 255, 127, 126, 255, 48, 49, 50,
    51, 52, 53, 54, 55, 56, 57, 255, 255, 255,
    255, 255, 255, 255, 65, 66, 67, 68, 69, 70,
    71, 72, 73, 74, 75, 76, 77, 78, 79, 80,
    81, 82, 83, 84, 85, 86, 87, 88, 89, 90,
    255, 255, 255, 255, 255, 48, 49, 50, 51, 52,
    53, 54, 55, 56, 57, 42, 43, 255, 45, 255,
    47, 255, 255, 255, 255, 255, 255, 255, 255, 255,
    255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
    255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
    255, 255, 255, 124, 255, 255, 255, 255, 255, 255,
    255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
    255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
    255, 255, 255, 255, 255, 59, 61, 44, 45, 46,
    47, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
    255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
    255, 255, 255, 255, 255, 255, 255, 255, 91, 92,
    93, 39, 255, 255, 255, 255, 255, 255, 255, 255, 255,
    255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
    255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
}

```

Translation table.

Translates Windows Virtual Keys to ATASCII codes doesn't handles all keys, to acces eg. small caps, you have to do it in the emulator's operating system (which isn't done by me :o)

Definition at line 25 of file perfdll.c.

5.16 usrmsgs.h File Reference

5.16.1 Detailed Description

Common file for all plugins. Here are defined all user messages used by the plugins. Also serves as a list of reserved message numbers

Definition in file [usrmsgs.h](#).

Defines

- `#define NMIMSG WM_USER + 0`
Message sent to the msg. queue when NMI interrupt occurs.
- `#define IRQMSG WM_USER + 1`
Message sent to the msg. queue when IRQ interrupt occurs.
- `#define RESETMSG WM_USER + 2`
Message sent to the msg. queue when RESET interrupt occurs.
- `#define SETFREQ WM_USER + 3`
lParam of the message is desired frequency in Hz
- `#define GETREGS WM_USER + 4`
lParam retrieves adress of [Regs](#) struct, where registers are stored. lParam should point to some UINT
- `#define ISRUNNING WM_USER + 5`
Returns TRUE if running otherwise FALSE.
- `#define ISTEP WM_USER + 6`
Executes one instruction. lParam should point to some UINT. lParam retrieves no. of cycles the instruction has taken.
- `#define IRQD WM_USER + 7`
A debug message, execute this interrupt immediately.
- `#define NMID WM_USER + 8`
A debug message, execute this interrupt immediately.
- `#define RESETD WM_USER + 9`
A debug message, execute this interrupt immediately.
- `#define WAITEND WM_USER + 10`
- `#define STARTCPU WM_USER + 11`
Begins cpu execution.
- `#define PAUSECPU WM_USER + 12`
Stops cpu execution.
- `#define CPURUNNING WM_USER + 13`

Sent by cpu plugin when cpu begins execution.

- **#define CPUSTOPPED WM_USER + 14**
Sent by cpu plugin when cpu stops.
- **#define STOPVIDEO WM_USER + 15**
Turns off the video chip.

5.16.2 Define Documentation

5.16.2.1 #define CPURUNNING WM_USER + 13

Sent by cpu plugin when cpu begins execution.

Definition at line 38 of file usrmsgs.h.

5.16.2.2 #define CPUSTOPPED WM_USER + 14

Sent by cpu plugin when cpu stops.

Definition at line 40 of file usrmsgs.h.

5.16.2.3 #define GETREGS WM_USER + 4

IParam retrieves address of [Regs](#) struct, where registers are stored. IParam should point to some UINT

Definition at line 20 of file usrmsgs.h.

5.16.2.4 #define IRQD WM_USER + 7

A debug message, execute this interrupt immediately.

Definition at line 26 of file usrmsgs.h.

5.16.2.5 #define IRQMSG WM_USER + 1

Message sent to the msg. queue when IRQ interrupt occurs.

Definition at line 14 of file usrmsgs.h.

5.16.2.6 #define ISRUNNING WM_USER + 5

Returns TRUE if running otherwise FALSE.

Definition at line 22 of file usrmsgs.h.

5.16.2.7 #define ISTEP WM_USER + 6

Executes one instruction. IParam should point to some UINT. IParam retrieves no. of cycles the instruction has taken.

Definition at line 24 of file usrmsgs.h.

5.16.2.8 **#define NMID WM_USER + 8**

A debug message, execute this interrupt immediately.

Definition at line 28 of file usrmsgs.h.

5.16.2.9 **#define NMIMSG WM_USER + 0**

Message sent to the msg. queue when NMI interrupt occurs.

Definition at line 12 of file usrmsgs.h.

5.16.2.10 **#define PAUSECPU WM_USER + 12**

Stops cpu execution.

Definition at line 36 of file usrmsgs.h.

5.16.2.11 **#define RESETD WM_USER + 9**

A debug message, execute this interrupt immediately.

Definition at line 30 of file usrmsgs.h.

5.16.2.12 **#define RESETMSG WM_USER + 2**

Message sent to the msg. queue when RESET interrupt occurs.

Definition at line 16 of file usrmsgs.h.

5.16.2.13 **#define SETFREQ WM_USER + 3**

lParam of the message is desired frequency in Hz

Definition at line 18 of file usrmsgs.h.

5.16.2.14 **#define STARTCPU WM_USER + 11**

Begins cpu execution.

Definition at line 34 of file usrmsgs.h.

5.16.2.15 **#define STOPVIDEO WM_USER + 15**

Turns off the video chip.

Definition at line 42 of file usrmsgs.h.

5.16.2.16 #define WAITEND WM_USER + 10

Waits until cpu stops if running and message PAUSECPU has been sent

Definition at line 32 of file usmsgs.h.

5.17 videodll.c File Reference

5.17.1 Detailed Description

A display plugin for emu6502. Reads video part of memory and displays what it contains. Also paints the text mode screen with given charset.

```
Definition in file videodll.c:#include "../dllkit/dllkit.h"
#include "../usrmsgs.h"
#include "resource.h"
#include <stdio.h>
#include <tchar.h>
```

Data Structures

- struct [b4iheader](#)
DIB header.

Timer defs

- #define [MYTIMER](#) 10001
ID of local timer.
- #define [PERIOD](#) 50
The period between refreshing screen in ms.

Screen defs

- #define [FWIDTH](#) 8
Pixels per character.
- #define [FHEIGHT](#) 8
Pixels per character.
- #define [YCOUNT](#) 32
Characters per screen.
- #define [XCOUNT](#) 32
Characters per screen.
- #define [WIDTH](#) FWIDTH * XCOUNT
Width of the screen.
- #define [HEIGHT](#) FHEIGHT * YCOUNT
Height of the screen.

IO addresses

- #define **GRMODE** IOLO + 0
- #define **LOADR** IOLO + 1
- #define **HIADR** IOLO + 2
- #define **LOCHR** IOLO + 3
- #define **HICHR** IOLO + 4
- #define **FGCOLR** IOLO + 5
- #define **FGCOLG** IOLO + 6
- #define **FGCOLB** IOLO + 7
- #define **BKCOLR** IOLO + 8
- #define **BKCOLG** IOLO + 9
- #define **BKCOLB** IOLO + 10
- #define **VIDEOON** IOLO + 14
- #define **COMMAND** IOLO + 15

IO registers - copied after COMMAND

- unsigned char volatile **_GrMode** = 0
- unsigned char volatile **_LoAdr** = 0x00
- unsigned char volatile **_HiAdr** = 0xA0
- unsigned char volatile **_LoChr** = 0x00
- unsigned char volatile **_HiChr** = 0xC0
- unsigned char volatile **_BkColR** = 0xFF
- unsigned char volatile **_BkColG** = 0xFF
- unsigned char volatile **_BkColB** = 0xFF
- unsigned char volatile **_FgColR** = 0x00
- unsigned char volatile **_FgColG** = 0x00
- unsigned char volatile **_FgColB** = 0x00
- unsigned char volatile **GrMode** = 0
- unsigned char volatile **LoAdr** = 0x00
- unsigned char volatile **HiAdr** = 0xA0
- unsigned char volatile **LoChr** = 0x00
- unsigned char volatile **HiChr** = 0xC0
- unsigned char volatile **VideoOn** = 0

Standard plugin variables

- HANDLE **hThisModule**
- MSGLIST **VideoMsgs**
- volatile BOOL **Running** = FALSE
- INITDLLSTRUCT **info**

Defines

- #define **ADR(X, Y)** (((X) << 8) + (Y))
- #define **IOSETGET(GVAR, SVAR)**

Functions

- void [InitGraphics](#) ()
Initializes DIB and logo structures.
- VOID CALLBACK [TimerProc](#) (HWND hwnd, UINT uMsg, UINT idEvent, DWORD dwTime)
Paints the screen every n ms if needed.
- void [Repaint](#) ()
Invalidates whole screen and evokes WM_PAINT.
- void [PaintScreen](#) (HDC hdc)
Paints whole screen.
- void [CopyChar](#) (char *dest, UINT adr, int x, int y, BOOL invert)
Copies one character to given coordinates.
- BOOL WINAPI [DllMain](#) (HANDLE hModule, DWORD ul_reason_for_call, LPVOID lpReserved)
- DLLEXPORT void __cdecl [InitDll](#) (INITDLLSTRUCT *ids, INITDLLINFO *idi)
Initializes dll data structures.
- DLLEXPORT void __cdecl [OnMessage](#) (MESSAGEDLLSTRUCT *mds)
Receives registered messages from emu6502 application.
- DLLEXPORT void __cdecl [IOFunc](#) (UINT adr, unsigned char *data, iomode mode)
IO function mapped into some address in memory.

Variables

- UINT [memStart](#) = 0xA000
address of videomem
- UINT [memChrStart](#) = 0xC000
address of charmap
- [b4iheader](#) [bGrInfo](#)
DIB header.
- char * [OffScreen](#)
An off-screen buffer for text mode.
- HBITMAP [hLogo](#)
Handle to the logo bitmap.
- HDC [hOffDC](#)
A compatible DC used for drawing the logo.

5.17.2 Define Documentation

5.17.2.1 #define FHEIGHT 8

Pixels per character.

Definition at line 31 of file videodll.c.

5.17.2.2 #define FWIDTH 8

Pixels per character.

Definition at line 29 of file videodll.c.

5.17.2.3 #define HEIGHT FHEIGHT * YCOUNT

Height of the screen.

Definition at line 39 of file videodll.c.

5.17.2.4 #define IOSETGET(GVAR, SVAR)

Value:

```
if (mode == IOREAD) { \
    *data = GVAR; \
} \
else { \
    SVAR = *data; \
}
```

Definition at line 289 of file videodll.c.

5.17.2.5 #define MYTIMER 10001

ID of local timer.

Definition at line 21 of file videodll.c.

5.17.2.6 #define PERIOD 50

The period between refreshing screen in ms.

Definition at line 23 of file videodll.c.

5.17.2.7 #define WIDTH FWIDTH * XCOUNT

Width of the screen.

Definition at line 37 of file videodll.c.

5.17.2.8 #define XCOUNT 32

Characters per screen.

Definition at line 35 of file videodll.c.

5.17.2.9 #define YCOUNT 32

Characters per screen.

Definition at line 33 of file videodll.c.

5.17.3 Function Documentation

5.17.3.1 void CopyChar (char * *dest*, UINT *adr*, int *x*, int *y*, BOOL *invert*)

Copies one character to given coordinates.

Uses character map stored in emulators memory

Parameters:

dest Address of text mode videomem

adr Address of the character

x X coordinate to paint the character onto

y Y coordinate to paint the character onto

invert Determines if the character bitmap should be inverted. If so, character no. - 128 is used.

Definition at line 392 of file videodll.c.

5.17.3.2 DLLEXPORT void __cdecl InitDll (INITDLLSTRUCT * *ids*, INITDLLINFO * *idi*)

Inits dll data structures.

Todo:

free VideoMsgs

Definition at line 175 of file videodll.c.

5.17.3.3 void InitGraphics ()

Inits DIB and logo structures.

Definition at line 203 of file videodll.c.

5.17.3.4 DLLEXPORT void __cdecl IOFunc (UINT *adr*, unsigned char * *data*, iomode *mode*)

IO function mapped into some adress in memory.

Called directly when accesing adress to which is this function bound. Always write this function.

Parameters:

adr Adress being accessed

data Byte being read/written

mode see *iomode*

Definition at line 297 of file videodll.c.

5.17.3.5 DLLEXPORT void _cdecl OnMessage (MESSAGEDLLSTRUCT * mds)

Receives registered messages from emu6502 application.

This function is called by emu6502 when some message event registered by *InitDll* occurs. Implement this function even when you don't want to catch any messages.

Parameters:

mds A *MESSAGEDLLSTRUCT* structure containing informations about incoming message.

See also:

[InitDll](#) , [MESSAGEDLLSTRUCT](#)

Definition at line 228 of file videodll.c.

5.17.3.6 void PaintScreen (HDC hdc)

Paints whole screen.

Called by WM_PAINT if the window needs repaint or every n ms when emulator is running For creating text mode screen calls *CopyChar*.

Parameters:

hdc DC of the canvas

See also:

[CopyChar](#)

Definition at line 365 of file videodll.c.

5.17.3.7 void Repaint ()

Invalidates whole screen and evokes WM_PAINT.

Definition at line 283 of file videodll.c.

5.17.3.8 VOID CALLBACK TimerProc (HWND hwnd, UINT uMsg, UINT idEvent, DWORD dwTime)

Paints the screen every n ms if needed.

Definition at line 271 of file videodll.c.

5.17.4 Variable Documentation

5.17.4.1 struct b4iheader bGrInfo

DIB header.

5.17.4.2 HBITMAP hLogo

Handle to the logo bitmap.

Definition at line 100 of file videodll.c.

5.17.4.3 HDC hOffDC

A compatible DC used for drawing the logo.

Definition at line 102 of file videodll.c.

5.17.4.4 UINT memChrStart = 0xC000

adress of charmap

Definition at line 87 of file videodll.c.

5.17.4.5 UINT memStart = 0xA000

adress of videomem

Definition at line 85 of file videodll.c.

5.17.4.6 char* OffScreen

An off-screen buffer for text mode.

Definition at line 97 of file videodll.c.

Chapter 6

Emu6502 Page Documentation

6.1 Todo List

Global **InstrStep**() Add undocumented instructions

Global **InitDll**(**INITDLLSTRUCT** *ids, **INITDLLINFO** *idf) free VideoMsgs

Global **WndProc**(**HWND** hWnd, **UINT** message, **WPARAM** wParam, **LPARAM** lParam) add TB-
STYLE.TOOLTIPS

Global **MYTIMER** Crashes - send message to msg. loop and stop

Index

- `_BkColB`
 - `videodll.c`, 98
 - `_BkColG`
 - `videodll.c`, 98
 - `_BkColR`
 - `videodll.c`, 98
 - `_FgColB`
 - `videodll.c`, 98
 - `_FgColG`
 - `videodll.c`, 98
 - `_FgColR`
 - `videodll.c`, 98
 - `_GrMode`
 - `videodll.c`, 98
 - `_HiAdr`
 - `videodll.c`, 98
 - `_HiChr`
 - `videodll.c`, 98
 - `_Instr`, 7
 - `aType`, 7
 - `len`, 7
 - `name`, 7
 - `_LoAdr`
 - `videodll.c`, 98
 - `_LoChr`
 - `videodll.c`, 98
 - A
 - `Regs`, 13
 - About
 - `emu6502.c`, 70
 - `emu6502.h`, 75
 - ABS
 - `cpu.c`, 18
 - ABSX
 - `cpu.c`, 18
 - ABSY
 - `cpu.c`, 19
 - ADC
 - `cpu.c`, 19
 - AddDllMsg
 - `emu6502.c`, 70
 - `emu6502.h`, 75
 - AddMessage
 - `dllkit.c`, 60
 - `dllkit.h`, 64
 - AddToPtrList
 - `emu6502.c`, 70
 - `emu6502.h`, 76
 - ADR
 - `cpu.c`, 19
 - `debugdll.c`, 52
 - `perifdll.c`, 91
 - `videodll.c`, 98
 - AdrType
 - `instructions.h`, 82
 - AND
 - `cpu.c`, 19
 - ASL
 - `cpu.c`, 20
 - ASLA
 - `cpu.c`, 20
 - aType
 - `_Instr`, 7
 - b4iheader, 8
 - `bmiColors`, 8
 - `bmiHeader`, 8
 - BCC
 - `cpu.c`, 20
 - BCS
 - `cpu.c`, 20
 - BEQ
 - `cpu.c`, 21
 - bGrInfo
 - `videodll.c`, 102
 - BIT
 - `cpu.c`, 21
 - BKCOLB
 - `videodll.c`, 98
 - BKCOLG
 - `videodll.c`, 98
 - BKCOLR
 - `videodll.c`, 98
 - BMI
 - `cpu.c`, 21
 - bmiColors
 - `b4iheader`, 8
 - bmiHeader
 - `b4iheader`, 8
-

BNE
 cpu.c, 22

BPL
 cpu.c, 22

BRK
 cpu.c, 23

BVC
 cpu.c, 23

BVS
 cpu.c, 23

CLC
 cpu.c, 24

CLD
 cpu.c, 24

CLFLAG
 cpu.c, 24

CLI
 cpu.c, 24

CLV
 cpu.c, 24

CMP
 cpu.c, 25

COMMAND
 videodll.c, 98

CONSABSX
 cpu.c, 25

CONSABSY
 cpu.c, 25

CONSINDY
 cpu.c, 25

CONSREL
 cpu.c, 26

CONSUME
 cpu.c, 26

ContExe
 cpudll.c, 44

CopyChar
 videodll.c, 101

cpu.c, 15
 ABS, 18
 ABSX, 18
 ABSY, 19
 ADC, 19
 ADR, 19
 AND, 19
 ASL, 20
 ASLA, 20
 BCC, 20
 BCS, 20
 BEQ, 21
 BIT, 21
 BMI, 21
 BNE, 22
 BPL, 22
 BRK, 23
 BVC, 23
 BVS, 23
 CLC, 24
 CLD, 24
 CLFLAG, 24
 CLI, 24
 CLV, 24
 CMP, 25
 CONSABSX, 25
 CONSABSY, 25
 CONSINDY, 25
 CONSREL, 26
 CONSUME, 26
 CPX, 26
 CPY, 26
 DEC, 26
 DEC2HEX, 27
 DEX, 27
 DEY, 27
 EOR, 27
 FALSE, 18
 FB, 27
 FC, 27
 FD, 28
 FI, 28
 FLAG, 28
 FN, 28
 FV, 28
 FZ, 28
 GETFLAG, 28
 GetRegs, 18
 HEX2DEC, 28
 HI, 28
 IMM, 29
 INC, 29
 INDX, 29
 INDY, 29
 InstrStep, 39
 INX, 29
 INY, 29
 Irq, 39
 IRQVKT, 30
 JMP, 30
 JMPADR, 30
 JMPIND, 30
 JSR, 30
 LDA, 30
 LDX, 31
 LDY, 31
 LO, 31
 LSR, 31
 LSRA, 31

- M2PC, 32
- Nmi, 39
- NMIVKT, 15
- NOP, 32
- ORA, 32
- PAGESIZE, 18
- PCSTEP, 32
- PHA, 32
- PHP, 32
- PLA, 33
- PLP, 33
- regs, 40
- REL, 33
- Reset, 39
- RESETVKT, 33
- ROL, 33
- ROLA, 34
- ROR, 34
- RORA, 34
- RTI, 34
- RTS, 35
- SBC, 35
- SEC, 35
- SED, 36
- SEI, 36
- SETFLAG, 36
- SETNZFLAGS, 36
- SetRegs, 18
- STA, 36
- STACKADD, 36
- STACKREMOVE, 37
- STX, 37
- STY, 37
- TAX, 37
- TAY, 37
- TRUE, 18
- TSX, 38
- TXA, 38
- TXS, 38
- TYA, 38
- ZP, 38
- ZPX, 38
- ZPY, 39
- cpu.h, 41
 - GetRegs, 41
 - InstrStep, 41
 - Irq, 42
 - memgetb, 41
 - memsetb, 41
 - Nmi, 42
 - Reset, 42
 - SetRegs, 41
 - UINT, 41
- cpudll.c, 43
 - ContExe, 44
 - CpuThreadFunc, 44
 - CpuThreadHandle, 46
 - CpuThreadID, 46
 - CreateCpuThread, 44
 - DllMain, 44
 - ExitCpuThread, 45
 - Freq, 46
 - Frms, 46
 - hThisModule, 44
 - info, 44
 - InitDll, 45
 - Interrupt, 46
 - IOFunc, 45
 - Msgs, 44
 - OnMessage, 45
 - Running, 46
 - secstart, 46
 - start, 46
 - ThreadState, 47
 - ticks, 47
 - total, 47
- CPURUNNING
 - usrmsgs.h, 94
- CPUSTOPPED
 - usrmsgs.h, 94
- CpuThreadFunc
 - cpudll.c, 44
- CpuThreadHandle
 - cpudll.c, 46
- CpuThreadID
 - cpudll.c, 46
- cpuuidll.c, 48
 - DllMain, 49
 - hCpuMenu, 50
 - hMenu, 50
 - hThisModule, 50
 - info, 50
 - InitDll, 49
 - IOFunc, 49
 - Msgs, 50
 - OnMessage, 49
 - Running, 49
 - SetStatusRunning, 50
 - SetStatusStopped, 50
- CPX
 - cpu.c, 26
- CPY
 - cpu.c, 26
- CR
 - emu6502.c, 68
- CreateCpuThread
 - cpudll.c, 44
- CreateMsgList

- dllkit.c, 60
- dllkit.h, 64
- CreateToolBar
 - emu6502.c, 70
 - emu6502.h, 76
- CSMemory
 - INITDLLSTRUCT, 10
 - mem.c, 85
 - mem.h, 89
- CTRLFLAG
 - perifdll.c, 91
- debugdll.c, 52
 - ADR, 52
 - DisAsm, 55
 - DllMain, 55
 - DumpMem, 55
 - DumpMemHex, 55
 - FB, 54
 - FC, 54
 - FD, 54
 - FI, 54
 - FN, 54
 - FV, 54
 - FZ, 54
 - GETFLAG, 52
 - hDebugMenu, 58
 - hMenu, 58
 - hThisModule, 54
 - info, 54
 - InitDll, 55
 - IOFunc, 56
 - mklpstr, 56
 - Msgs, 54
 - ofile, 56
 - OnMessage, 56
 - ofile, 57
 - print, 57
 - printact, 57
 - printfil, 57
 - printhelp, 57
 - printregs, 57
 - rhex, 58
 - rinterval, 58
 - rline, 58
 - SetReg, 58
 - StartDebug, 58
- DEC
 - cpu.c, 26
- DEC2HEX
 - cpu.c, 27
- DEX
 - cpu.c, 27
- DEY
 - cpu.c, 27
- DHInstList
 - emu6502.c, 73
- DisAsm
 - debugdll.c, 55
- DLLEXPORT
 - dllkit.h, 62
- dllkit.c, 60
 - AddMessage, 60
 - CreateMsgList, 60
 - FreeMsgList, 60
- dllkit.h, 62
 - AddMessage, 64
 - CreateMsgList, 64
 - DLLEXPORT, 62
 - ENDOFMSGLIST, 63
 - FreeMsgList, 64
 - InitDll, 65
 - IOFunc, 65
 - IOHI, 63
 - IOLO, 63
 - iomode, 64
 - IRQID, 62
 - ISVBI, 62
 - MSGLIST, 64
 - NMIID, 63
 - OnMessage, 65
 - RESETID, 62
- DllMain
 - cpudll.c, 44
 - cpuuidll.c, 49
 - debugdll.c, 55
 - genericdll.c, 80
 - perifdll.c, 91
 - videodll.c, 99
- DllMsgList
 - emu6502.h, 75
- dlluser.h, 66
 - ENDOFMSGLIST, 67
 - iomode, 67
 - MSGLIST, 67
 - pForwardDllMsgProc, 67
 - pInitDllProc, 67
 - pIOFunc, 67
- DMList
 - emu6502.c, 73
- DumpMem
 - debugdll.c, 55
- DumpMemHex
 - debugdll.c, 55
- emu6502.c, 68
 - About, 70
 - AddDllMsg, 70

- AddToPtrList, 70
- CR, 68
- CreateToolBar, 70
- DHInstList, 73
- DMList, 73
- error, 70
- ForwardDllMessage, 71
- GetNextStrToken, 71
- hInst, 73
- hMainWnd, 73
- hToolBar, 73
- InitInstance, 71
- LF, 68
- LoadMemFile, 71
- LoadPlugin, 72
- LoadPlugins, 72
- MAX_LOADSTRING, 68
- MyRegisterClass, 72
- szTitle, 69
- szWindowClass, 69
- UnloadPlugins, 69
- WIN32_LEAN_AND_MEAN, 68
- WinMain, 72
- WndProc, 72
- emu6502.h, 74
 - About, 75
 - AddDllMsg, 75
 - AddToPtrList, 76
 - CreateToolBar, 76
 - DllMsgList, 75
 - error, 76
 - ForwardDllMessage, 76
 - GetNextStrToken, 77
 - InitInstance, 77
 - LoadMemFile, 77
 - LoadPlugin, 77
 - LoadPlugins, 78
 - MyRegisterClass, 78
 - PtrList, 75
 - UnloadPlugins, 75
 - WinMain, 78
 - WndProc, 78
- ENDOFMSGLIST
 - dllkit.h, 63
 - dlluser.h, 67
- EOR
 - cpu.c, 27
- error
 - emu6502.c, 70
 - emu6502.h, 76
- ExitCpuThread
 - cpudll.c, 45
- FALSE
 - cpu.c, 18
 - mem.h, 87
- FB
 - cpu.c, 27
 - debugdll.c, 54
- FC
 - cpu.c, 27
 - debugdll.c, 54
- FD
 - cpu.c, 28
 - debugdll.c, 54
- FGCOLB
 - videodll.c, 98
- FGCOLG
 - videodll.c, 98
- FGCOLR
 - videodll.c, 98
- FHEIGHT
 - videodll.c, 100
- FI
 - cpu.c, 28
 - debugdll.c, 54
- FLAG
 - cpu.c, 28
- FN
 - cpu.c, 28
 - debugdll.c, 54
- ForwardDllMessage
 - emu6502.c, 71
 - emu6502.h, 76
- FreeMsgList
 - dllkit.c, 60
 - dllkit.h, 64
- Freq
 - cpudll.c, 46
- Frms
 - cpudll.c, 46
- FV
 - cpu.c, 28
 - debugdll.c, 54
- FWIDTH
 - videodll.c, 100
- FZ
 - cpu.c, 28
 - debugdll.c, 54
- genericdll.c, 79
 - DllMain, 80
 - hThisModule, 80
 - info, 80
 - InitDll, 80
 - IOFunc, 80
 - IOSETGET, 79
 - Msgs, 80

- OnMessage, 80
- GETFLAG
 - cpu.c, 28
 - debugdll.c, 52
- GetNextStrToken
 - emu6502.c, 71
 - emu6502.h, 77
- GETREGS
 - usrmsgs.h, 94
- GetRegs
 - cpu.c, 18
 - cpu.h, 41
- GRMODE
 - videodll.c, 98
- GrMode
 - videodll.c, 98
- hCpuMenu
 - cpuuidll.c, 50
- hDebugMenu
 - debugdll.c, 58
- HEIGHT
 - videodll.c, 100
- HEX2DEC
 - cpu.c, 28
- HI
 - cpu.c, 28
- HIADR
 - videodll.c, 98
- HiAdr
 - videodll.c, 98
- HICHR
 - videodll.c, 98
- HiChr
 - videodll.c, 98
- hInst
 - emu6502.c, 73
 - INITDLLSTRUCT, 10
- hLogo
 - videodll.c, 102
- hMainWnd
 - emu6502.c, 73
- hMenu
 - cpuuidll.c, 50
 - debugdll.c, 58
- hOffDC
 - videodll.c, 103
- hTb
 - INITDLLSTRUCT, 10
- hThisModule
 - cpudll.c, 44
 - cpuuidll.c, 50
 - debugdll.c, 54
 - genericdll.c, 80
 - perifdll.c, 91
 - videodll.c, 98
- hToolBar
 - emu6502.c, 73
- hWnd
 - INITDLLSTRUCT, 10
- IMM
 - cpu.c, 29
- INC
 - cpu.c, 29
- INDX
 - cpu.c, 29
- INDY
 - cpu.c, 29
- info
 - cpudll.c, 44
 - cpuuidll.c, 50
 - debugdll.c, 54
 - genericdll.c, 80
 - perifdll.c, 91
 - videodll.c, 98
- InitDll
 - cpudll.c, 45
 - cpuuidll.c, 49
 - debugdll.c, 55
 - dllkit.h, 65
 - genericdll.c, 80
 - perifdll.c, 92
 - videodll.c, 101
- INITDLLINFO, 9
 - iohi, 9
 - iolo, 9
 - msglist, 9
- INITDLLSTRUCT, 10
 - CSMemory, 10
 - hInst, 10
 - hTb, 10
 - hWnd, 10
 - memgetb, 10
 - memory, 11
 - memsetb, 11
 - optional, 11
 - rcPaint, 11
- InitGraphics
 - videodll.c, 101
- InitInstance
 - emu6502.c, 71
 - emu6502.h, 77
- InitMem
 - mem.c, 84
 - mem.h, 88
- Instr
 - instructions.h, 82

- InstrStep
 - cpu.c, 39
 - cpu.h, 41
- INSTRUCTIONS
 - instructions.h, 83
- instructions.h, 82
 - AdrType, 82
 - Instr, 82
- INSTRUCTIONS, 83
- Interrupt
 - cpudll.c, 46
- INX
 - cpu.c, 29
- INY
 - cpu.c, 29
- IOFunc
 - cpudll.c, 45
 - cpuuidll.c, 49
 - debugdll.c, 56
 - dllkit.h, 65
 - genericdll.c, 80
 - perifdll.c, 91
 - videodll.c, 101
- IOHI
 - dllkit.h, 63
 - mem.h, 88
- iohi
 - INITDLLINFO, 9
- IOLO
 - dllkit.h, 63
 - mem.h, 88
- iolo
 - INITDLLINFO, 9
- iomode
 - dllkit.h, 64
 - dlluser.h, 67
- IOSETGET
 - genericdll.c, 79
 - perifdll.c, 91
 - videodll.c, 100
- Irq
 - cpu.c, 39
 - cpu.h, 42
- IRQD
 - usrmsgs.h, 94
- IRQID
 - dllkit.h, 62
- IRQMSG
 - usrmsgs.h, 94
- IRQVKT
 - cpu.c, 30
- ISRUNNING
 - usrmsgs.h, 94
- ISTEP
 - usrmsgs.h, 94
- ISVBI
 - dllkit.h, 62
- JMP
 - cpu.c, 30
- JMPADR
 - cpu.c, 30
- JMPIND
 - cpu.c, 30
- JSR
 - cpu.c, 30
- KbMap
 - perifdll.c, 92
- KBSTAT
 - perifdll.c, 91
- KbState
 - perifdll.c, 91
- Key
 - perifdll.c, 91
- LDA
 - cpu.c, 30
- LDX
 - cpu.c, 31
- LDY
 - cpu.c, 31
- len
 - .Instr, 7
- LF
 - emu6502.c, 68
- LO
 - cpu.c, 31
- LoadMemFile
 - emu6502.c, 71
 - emu6502.h, 77
- LoadPlugin
 - emu6502.c, 72
 - emu6502.h, 77
- LoadPlugins
 - emu6502.c, 72
 - emu6502.h, 78
- LOADR
 - videodll.c, 98
- LoAdr
 - videodll.c, 98
- LOCHR
 - videodll.c, 98
- LoChr
 - videodll.c, 98
- lParam
 - MESSAGEDLLSTRUCT, 12
- LSR

- cpu.c, 31
- LSRA
 - cpu.c, 31
- M2PC
 - cpu.c, 32
- MAX_LOADSTRING
 - emu6502.c, 68
- mem.c, 84
 - CSMemory, 85
 - InitMem, 84
 - memgetb, 85
 - memio, 85
 - memory, 86
 - memsetb, 85
 - SetIOFunc, 85
 - SetMemory, 85
- mem.h, 87
 - CSMemory, 89
 - FALSE, 87
 - InitMem, 88
 - IOHI, 88
 - IOLO, 88
 - memgetb, 88
 - memio, 89
 - memory, 90
 - memsetb, 89
 - MEMSIZE, 88
 - PAGECOUNT, 88
 - PAGESIZE, 88
 - SetIOFunc, 89
 - SetMemory, 89
 - TRUE, 87
 - UINT, 87
- memChrStart
 - videodll.c, 103
- memgetb
 - cpu.h, 41
 - INITDLLSTRUCT, 10
 - mem.c, 85
 - mem.h, 88
- memio
 - mem.c, 85
 - mem.h, 89
- memory
 - INITDLLSTRUCT, 11
 - mem.c, 86
 - mem.h, 90
- memsetb
 - cpu.h, 41
 - INITDLLSTRUCT, 11
 - mem.c, 85
 - mem.h, 89
- MEMSIZE
 - mem.h, 88
- memStart
 - videodll.c, 103
- message
 - MESSAGEDLLSTRUCT, 12
- MESSAGEDLLSTRUCT, 12
 - IParam, 12
 - message, 12
 - wParam, 12
- mklpstr
 - debugdll.c, 56
- MSGLIST
 - dllkit.h, 64
 - dlluser.h, 67
- msglist
 - INITDLLINFO, 9
- Msgs
 - cpudll.c, 44
 - cpuuidll.c, 50
 - debugdll.c, 54
 - genericdll.c, 80
 - perifdll.c, 91
- MyRegisterClass
 - emu6502.c, 72
 - emu6502.h, 78
- MYTIMER
 - videodll.c, 100
- name
 - _Instr, 7
- Nmi
 - cpu.c, 39
 - cpu.h, 42
- NMID
 - usrmsgs.h, 95
- NMIID
 - dllkit.h, 63
- NMIMSG
 - usrmsgs.h, 95
- NMIVKT
 - cpu.c, 15
- NOP
 - cpu.c, 32
- OffScreen
 - videodll.c, 103
- ofile
 - debugdll.c, 56
- OnMessage
 - cpudll.c, 45
 - cpuuidll.c, 49
 - debugdll.c, 56
 - dllkit.h, 65
 - genericdll.c, 80

- perifdll.c, 91
- videodll.c, 102
- optional
 - INITDLLSTRUCT, 11
- ORA
 - cpu.c, 32
- orfile
 - debugdll.c, 57
- P
 - Regs, 13
- PAGECOUNT
 - mem.h, 88
- PAGESIZE
 - cpu.c, 18
 - mem.h, 88
- PaintScreen
 - videodll.c, 102
- PAUSECPU
 - usrmsgs.h, 95
- PC
 - Regs, 13
- PCSTEP
 - cpu.c, 32
- perifdll.c, 91
 - ADR, 91
 - CTRLFLAG, 91
 - DllMain, 91
 - hThisModule, 91
 - info, 91
 - InitDll, 92
 - IOFunc, 91
 - IOSETGET, 91
 - KbMap, 92
 - KBSTAT, 91
 - KbState, 91
 - Key, 91
 - Msgs, 91
 - OnMessage, 91
 - RANDOM, 91
 - SHIFTFLAG, 91
 - SHIFTSTAT, 91
- PERIOD
 - videodll.c, 100
- pForwardDllMsgProc
 - dlluser.h, 67
- PHA
 - cpu.c, 32
- PHP
 - cpu.c, 32
- pInitDllProc
 - dlluser.h, 67
- pIOFunc
 - dlluser.h, 67
- PLA
 - cpu.c, 33
- PLP
 - cpu.c, 33
- print
 - debugdll.c, 57
- printact
 - debugdll.c, 57
- printfil
 - debugdll.c, 57
- printhelp
 - debugdll.c, 57
- printregs
 - debugdll.c, 57
- PtrList
 - emu6502.h, 75
- RANDOM
 - perifdll.c, 91
- rcPaint
 - INITDLLSTRUCT, 11
- Regs, 13
 - A, 13
 - P, 13
 - PC, 13
 - S, 13
 - X, 13
 - Y, 13
- regs
 - cpu.c, 40
- REL
 - cpu.c, 33
- Repaint
 - videodll.c, 102
- Reset
 - cpu.c, 39
 - cpu.h, 42
- RESETD
 - usrmsgs.h, 95
- RESETID
 - dllkit.h, 62
- RESETMSG
 - usrmsgs.h, 95
- RESETVKT
 - cpu.c, 33
- rhex
 - debugdll.c, 58
- rinterval
 - debugdll.c, 58
- rline
 - debugdll.c, 58
- ROL
 - cpu.c, 33
- ROLA

- cpu.c, 34
- ROR
 - cpu.c, 34
- RORA
 - cpu.c, 34
- RTI
 - cpu.c, 34
- RTS
 - cpu.c, 35
- Running
 - cpudll.c, 46
 - cpuuidll.c, 49
 - videodll.c, 98
- S
 - Regs, 13
- SBC
 - cpu.c, 35
- SEC
 - cpu.c, 35
- secstart
 - cpudll.c, 46
- SED
 - cpu.c, 36
- SEI
 - cpu.c, 36
- SETFLAG
 - cpu.c, 36
- SETFREQ
 - usrmsgs.h, 95
- SetIOFunc
 - mem.c, 85
 - mem.h, 89
- SetMemory
 - mem.c, 85
 - mem.h, 89
- SETNZFLAGS
 - cpu.c, 36
- SetReg
 - debugdll.c, 58
- SetRegs
 - cpu.c, 18
 - cpu.h, 41
- SetStatusRunning
 - cpuuidll.c, 50
- SetStatusStopped
 - cpuuidll.c, 50
- SHIFTFLAG
 - perifdll.c, 91
- SHIFTSTAT
 - perifdll.c, 91
- STA
 - cpu.c, 36
- STACKADD
 - cpu.c, 36
- STACKREMOVE
 - cpu.c, 37
- start
 - cpudll.c, 46
- STARTCPU
 - usrmsgs.h, 95
- StartDebug
 - debugdll.c, 58
- STOPVIDEO
 - usrmsgs.h, 95
- STX
 - cpu.c, 37
- STY
 - cpu.c, 37
- szTitle
 - emu6502.c, 69
- szWindowClass
 - emu6502.c, 69
- TAX
 - cpu.c, 37
- TAY
 - cpu.c, 37
- ThreadState
 - cpudll.c, 47
- ticks
 - cpudll.c, 47
- TimerProc
 - videodll.c, 102
- total
 - cpudll.c, 47
- TRUE
 - cpu.c, 18
 - mem.h, 87
- TSX
 - cpu.c, 38
- TXA
 - cpu.c, 38
- TXS
 - cpu.c, 38
- TYA
 - cpu.c, 38
- UINT
 - cpu.h, 41
 - mem.h, 87
- UnloadPlugins
 - emu6502.c, 69
 - emu6502.h, 75
- usrmsgs.h, 93
 - CPURUNNING, 94
 - CPUSTOPPED, 94
 - GETREGS, 94

- IRQD, 94
- IRQMSG, 94
- ISRUNNING, 94
- ISTEP, 94
- NMID, 95
- NMIMSG, 95
- PAUSECPU, 95
- RESETD, 95
- RESETMSG, 95
- SETFREQ, 95
- STARTCPU, 95
- STOPVIDEO, 95
- WAITEND, 95
- videodll.c, 97
 - _BkColB, 98
 - _BkColG, 98
 - _BkColR, 98
 - _FgColB, 98
 - _FgColG, 98
 - _FgColR, 98
 - _GrMode, 98
 - _HiAdr, 98
 - _HiChr, 98
 - _LoAdr, 98
 - _LoChr, 98
 - ADR, 98
 - bGrInfo, 102
 - BKCOLB, 98
 - BKCOLG, 98
 - BKCOLR, 98
 - COMMAND, 98
 - CopyChar, 101
 - DllMain, 99
 - FGCOLB, 98
 - FGCOLG, 98
 - FGCOLR, 98
 - FHEIGHT, 100
 - FWIDTH, 100
 - GRMODE, 98
 - GrMode, 98
 - HEIGHT, 100
 - HIADR, 98
 - HiAdr, 98
 - HICHR, 98
 - HiChr, 98
 - hLogo, 102
 - hOffDC, 103
 - hThisModule, 98
 - info, 98
 - InitDll, 101
 - InitGraphics, 101
 - IOFunc, 101
 - IOSETGET, 100
 - LOADR, 98
 - LoAdr, 98
 - LOCHR, 98
 - LoChr, 98
 - memChrStart, 103
 - memStart, 103
 - MYTIMER, 100
 - OffScreen, 103
 - OnMessage, 102
 - PaintScreen, 102
 - PERIOD, 100
 - Repaint, 102
 - Running, 98
 - TimerProc, 102
 - VideoMsgs, 98
 - VIDEOON, 98
 - VideoOn, 98
 - WIDTH, 100
 - XCOUNT, 100
 - YCOUNT, 101
 - VideoMsgs
 - videodll.c, 98
 - VIDEOON
 - videodll.c, 98
 - VideoOn
 - videodll.c, 98
 - WAITEND
 - usrmsgs.h, 95
 - WIDTH
 - videodll.c, 100
 - WIN32_LEAN_AND_MEAN
 - emu6502.c, 68
 - WinMain
 - emu6502.c, 72
 - emu6502.h, 78
 - WndProc
 - emu6502.c, 72
 - emu6502.h, 78
 - wParam
 - MESSAGEDLLSTRUCT, 12
 - X
 - Regs, 13
 - XCOUNT
 - videodll.c, 100
 - Y
 - Regs, 13
 - YCOUNT
 - videodll.c, 101
 - ZP
 - cpu.c, 38

ZPX

cpu.c, [38](#)

ZPY

cpu.c, [39](#)