

**680x0**

<b>COLLABORATORS</b>
----------------------

	<i>TITLE :</i> 680x0		
<i>ACTION</i>	<i>NAME</i>	<i>DATE</i>	<i>SIGNATURE</i>
WRITTEN BY		August 9, 2024	

<b>REVISION HISTORY</b>
-------------------------

NUMBER	DATE	DESCRIPTION	NAME

# Contents

<b>1</b>	<b>680x0</b>	<b>1</b>
1.1	680x0.doc . . . . .	1
1.2	680x0.library/--Background-- . . . . .	1
1.3	680x0.library/CPUType . . . . .	1
1.4	680x0.library/FPUType . . . . .	2
1.5	680x0.library/MMUType . . . . .	3
1.6	680x0.library/SetFPUExceptions . . . . .	4

# Chapter 1

## 680x0

### 1.1 680x0.doc

```
--Background--  
CPUType()  
FPUType()  
MMUType()  
SetFPUEXceptions()
```

### 1.2 680x0.library/--Background--

#### PURPOSE

This library is the generic processor support library for all Amigas and all CPU types. Its purpose is to determinate which CPU, FPU and MMU type is installed, and to load the processor specific support code in form of the 68000.library, the 68010.library, the 68020.library, the 68030.library, the 68040.library or the 68060.library. It does not include CPU specific code instead, but re-routes all its calls to the processor specific library to allow CPU/FPU/MMU independent code. It loads also the mmu.library on startup to install the MMU specific functions.

This library provides user-callable functions to determinate the CPU/FPU and MMU type available. The results are more precise than those that can be found in SysBase->AttnFlags. Furthermore, the library controls the FPU exceptions.

The 680x0.library is usually loaded by "SetPatch" on startup and remains in the system. It then checks the available hardware and loads the CPU and MMU specific drivers.

### 1.3 680x0.library/CPUType

---

**NAME**

CPUType - return information about the available CPU

**SYNOPSIS**

```
cpu = CPUType ( );
```

```
char CPUType ( void );
```

**FUNCTION**

This function returns a one-bit identifier about the available CPU. It currently knows the 68000, 68010, 68020, 68030, 68040 and 68060.

**INPUTS**

none.

**RESULTS**

The CPU type found. One of the following identifiers is returned:

```
CPUTYPE_68000      /* a plain 68000*/
CPUTYPE_68010      /* a 68010 */
CPUTYPE_68020      /* a 68020 */
CPUTYPE_68030      /* a 68030 */
CPUTYPE_68040      /* a 68040 */
CPUTYPE_68060      /* a 68060 */
```

**NOTES**

Unlike some third-party libraries, this result code will be correct even for the 68060.

**BUGS**

Does currently not support a PPC.

**SEE ALSO**

libraries/680x0.h, exec/execbase.h

## 1.4 680x0.library/FPUType

**NAME**

FPUType - return information about the available FPU

**SYNOPSIS**

```
fpu = FPUType ( );
```

```
char FPUType ( void );
```

**FUNCTION**

This function returns a one-bit identifier of the available FPU. It currently knows the 68881, 68882 and the build-in FPUs of the 68040 and 68060.

**INPUTS**

none.

---

## RESULTS

The FPU type found. One of the following identifiers is returned:

```
FPUTYPE_NONE      /* no FPU available */
FPUTYPE_68881     /* the 68881 external FPU */
FPUTYPE_68882     /* the advanced edition of this chip */
FPUTYPE_68040     /* the 68040 buildin FPU */
FPUTYPE_68060     /* the 68060 buildin FPU */
```

## NOTES

This function works differently than SysBase->AttnFlags. It will return FPUTYPE\_68040 or FPUTYPE\_68060 if a 68040 or 68060 FPU is in the system, regardless whether the processor support code has been loaded and the 68881/68882 instructions can be emulated in software. This should be checked in SysBase->AttnFlags, if required.

## BUGS

Does currently not support a PPC.

## SEE ALSO

libraries/680x0.h, exec/execbase.h

## 1.5 680x0.library/MMUType

## NAME

MMUType - return the type of the MMU available in the system.

## SYNOPSIS

```
mmu = GetMMUType( );
```

```
char GetMMUType( void );
```

## FUNCTION

Returns an identifier for the MMU available in the system or NUL in case no MMU is installed.

## INPUTS

none.

## RETURNS

a character identifying the MMU type:

```
MUTYPE_NONE      no working MMU detected.
MUTYPE_68851     a 68020 system with an external 68851
                  MMU.
MUTYPE_68030     a 68030 MMU.
MUTYPE_68040     the internal 68040 MMU.
MUTYPE_68060     the 68060 MMU.
```

## NOTES

This call is directly re-routed to the mmu.library GetMMUType() call. If the mmu.library is not available, it will return MUTYPE\_NONE.

The mmu library is smart enough to detect EC processors without a working MMU, but the library does not detect multiple CPUs in the system. (How?)

BUGS

SEE ALSO

libraries/680x0.h, mmu/mmubase.h

## 1.6 680x0.library/SetFPUEExceptions

NAME

SetFPUEExceptions - control the generation of FPU exceptions

SYNOPSIS

```
oldflags = SetFPUEExceptions ( flags , mask );
d0          d0          d1
```

```
ULONG SetFPUEExceptions ( ULONG , ULONG );
```

FUNCTION

This function disables or enables various exceptions a MC68K FPU might generate. The purpose of this function is mainly to enable workarounds for badly written software.

INPUTS

flags - A ULONG bit mask of the exceptions to disable. A set bit disables the corresponding exception.  
mask - A mask longword of the flags that are to be changed. A set bit indicates that the corresponding bit in flags mask is valid.

The following bits are currently defined:

FPUCtrlB_BSUN	0L	Disable the branch or set on unordered exception.
FPUCtrlB_INEX	1L	Disable the inexact result exception.
FPUCtrlB_DIVZ	2L	Disable the divide by zero exception.
FPUCtrlB_UNFL	3L	Disable the underflow exception.
FPUCtrlB_OVFL	4L	Disable the overflow exception.
FPUCtrlB_SNAN	5L	Disable the signalling NAN exception.
FPUCtrlB_OPERR	6L	Disable the operand error.

RESULTS

the old settings of the exception mask.

NOTES

This function is directly re-routed to the CPU specific driver, i.e. either the 68040, the 68060, the 68020 or the 68030.library. It does not touch the FPU itself. If the driver is not available, or does not allow the change the FPU exception mask, this procedure does nothing and returns "0".

BUGS

SEE ALSO  
libraries/68040.h, libraries/680x0.h,  
the Motorola 68040 manual.