

# Technische Manual: ML64 Epoch DLL voor Rocket COBOL 11.0 en Intel C

## Technische Manual

ML64 → DLL → Rocket COBOL 11.0 → Intel C

Volledige Implementatie- en Integratiehandleiding

---

### 1. Overzicht

Deze technische handleiding beschrijft de volledige implementatieketen voor het bouwen van een Windows x64 DLL met een ML64-assembleroutine die Unix-epochseconden retourneert, en het gebruik daarvan binnen zowel Rocket COBOL 11.0 als Intel C. De inhoud is modulair, reproduceerbaar en vrij van legacy-code.

---

### 2. ML64 Assembly Routine (epoch.asm)

De volgende ML64-routine gebruikt de Windows API `GetSystemTimeAsFileTime` om de huidige UTC-tijd op te halen, converteert deze naar Unix-epochseconden en retourneert het resultaat in register **RAX**.

```

; epoch.asm - get_epoch_seconds in ML64 (x64)
; __int64 get_epoch_seconds(void);

OPTION PROLOGUE:NONE
OPTION EPILOGUE:NONE

EXTERN GetSystemTimeAsFileTime:PROC
PUBLIC get_epoch_seconds

.data
; 1164447360000000000 = verschil tussen 1601 en 1970 in 100-ns ticks
EpochOffsetLow QWORD 0xD53E8000
EpochOffsetHigh QWORD 0x019DB1DE

.code

get_epoch_seconds PROC
sub rsp, 48

lea rcx, [rsp+16]
call GetSystemTimeAsFileTime

mov eax, DWORD PTR [rsp+16]
mov edx, DWORD PTR [rsp+20]

mov r8, QWORD PTR EpochOffsetLow
mov r9, QWORD PTR EpochOffsetHigh

sub rax, r8
sbb rdx, r9

mov ecx, 10000000
div ecx

add rsp, 48
ret
get_epoch_seconds ENDP

END

```

### 3. DLL Export Definitie (epochdll.def)

```
LIBRARY "epochdll"  
EXPORTS  
get_epoch_seconds
```

---

## 4. Buildproces

Alle stappen worden uitgevoerd in een **x64 Developer Command Prompt**.

### 4.1 Assemblage

```
ml64 /c /Fo epoch.obj epoch.asm
```

### 4.2 Linken naar DLL (inclusief import library voor C)

```
link /DLL /NOENTRY epoch.obj kernel32.lib ^  
/DEF:epochdll.def ^  
/OUT:epochdll.dll ^  
/IMPLIB:epochdll.lib
```

### 4.3 Validatie

```
dumpbin /EXPORTS epochdll.dll
```

De export get\_epoch\_seconds moet zichtbaar zijn.

---

## 5. Rocket COBOL 11.0 Integratie

Onderstaand COBOL-programma roept de DLL aan en ontvangt de 64-bit epochwaarde via RETURNING.

```
IDENTIFICATION DIVISION.  
PROGRAM-ID. TESTEPOCH.  
  
DATA DIVISION.  
WORKING-STORAGE SECTION.  
01 EPOCH-SECONDS PIC S9(18) COMP-5.  
  
PROCEDURE DIVISION.  
CALL "get_epoch_seconds"  
RETURNING EPOCH-SECONDS  
  
DISPLAY "Unix epoch seconds: " EPOCH-SECONDS  
  
STOP RUN.
```

## Belangrijke punten

- COMP-5 garandeert een native 64-bit integer.
  - De DLL moet zich in dezelfde directory bevinden als de COBOL-executable, of in een map die in PATH staat.
- 

## 6. Intel C Integratie

Dezelfde DLL kan direct worden gebruikt vanuit Intel C of elke MSVC-compatibele C-compiler.

### 6.1 C Header (epochdll.h)

```
#ifndef EPOCHDLL_H
#define EPOCHDLL_H

#ifdef __cplusplus
extern "C" {
#endif

__int64 __cdecl get_epoch_seconds(void);

#ifdef __cplusplus
}
#endif

#endif /* EPOCHDLL_H */
```

### 6.2 Testprogramma in C (test\_epoch.c)

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

int main(void)
{
    __int64 epoch = get_epoch_seconds();

    printf("Unix epoch seconds (from DLL): %lld\n", (long long)epoch);

    return 0;
}
```

### 6.3 Compileren en linken met Intel C

```
icl /c test_epoch.c
link test_epoch.obj epochdll.lib /OUT:test_epoch.exe
```

## 6.4 Compileren met MSVC

```
cl /c test_epoch.c  
link test_epoch.obj epochdll.lib /OUT:test_epoch.exe
```

---

## 7. Uitvoeren en Controle

### 7.1 COBOL

```
TESTEPOCH.exe
```

### 7.2 Intel C

```
test_epoch.exe
```

Beide programma's moeten vergelijkbare epochwaarden tonen.

---

## 8. Mogelijke Uitbreidingen

- Toevoegen van milliseconden of nanoseconden.
  - Exporteren van datum/tijd als geformatteerde string.
  - Meerdere exports binnen dezelfde DLL.
- 

## 9. Conclusie

Deze handleiding biedt een volledige, schone en moderne workflow voor het bouwen van een ML64-gebaseerde DLL en het integreren ervan in zowel Rocket COBOL 11.0 als Intel C. Alle stappen zijn direct toepasbaar in een professionele ontwikkelomgeving.