SYS TEC ELECTRONIC

Flying Master Add-on

Software Manual

Auflage September 2013

Systemhaus für verteilte Automatisierung

Status / Änderungen

Status: Entwurf

Datum/ Version	Abschnitt	Änderung	Editor
30.09.2013/ 01		Erstanlage	Dietzsch

Im Buch verwendete Bezeichnungen für Erzeugnisse, die zugleich ein eingetragenes Warenzeichen darstellen, wurden nicht besonders gekennzeichnet. Das Fehlen der © Markierung ist demzufolge nicht gleichbedeutend mit der Tatsache, dass die Bezeichnung als freier Warenname gilt. Ebenso wenig kann anhand der verwendeten Bezeichnung auf eventuell vorliegende Patente oder einen Gebrauchsmusterschutz geschlossen werden.

Die Informationen in diesem Handbuch wurden sorgfältig überprüft und können als zutreffend angenommen werden. Dennoch sei ausdrücklich darauf verwiesen, dass die Firma SYS TEC electronic GmbH weder eine Garantie noch die juristische Verantwortung oder irgendeine Haftung für Folgeschäden übernimmt, die auf den Gebrauch oder den Inhalt dieses Handbuches zurückzuführen sind. Die in diesem Handbuch enthaltenen Angaben können ohne vorherige Ankündigung geändert werden. Die Firma SYS TEC electronic GmbH geht damit keinerlei Verpflichtungen ein.

Ferner sei ausdrücklich darauf verwiesen, dass SYS TEC electronic GmbH weder eine Garantie noch die juristische Verantwortung oder irgendeine Haftung für Folgeschäden übernimmt, die auf falschen Gebrauch oder falschen Einsatz der Hard- bzw. Software zurückzuführen sind. Ebenso können ohne vorherige Ankündigung Layout oder Design der Hardware geändert werden. SYS TEC electronic GmbH geht damit keinerlei Verpflichtungen ein.

Kontakt	**Direkt**	**Ihr Lokaler Distributor**
Adresse:	SYS TEC electronic GmbH Am Windrad 2 D-08468 Heinsdorfergrund GERMANY	Sie finden eine Liste unserer Distributoren unter http://www.systec-electronic.com/distributors
Angebots-Hotline:	+49 3765 38600-2110 info@systec-electronic.com	
Technische Hotline:	+49 3765 38600-2140 support@systec-electronic.com	
Fax:	+49 3765 38600-4100	
Webseite:	http://www.systec-electronic.com	

01. Auflage Oktober 2013

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Referenzierte Dokumente

/1/ CANopen User Manual L-1020, SYS TEC electronic GmbH

/2/ CANopen Manager Software Manual L-1085, SYS TEC electronic GmbH

/3/ CiA Draft Standard Proposal 302, Part 2 Multi-level networking, © CAN in Automation (CiA) e. V.

Abkürzungen

API	Application Programming Interface
CAN	Controller Area Network
CAN-ID	CAN Identifier, Arbitrierungsfeld des CAN-Telegramms. Es muss sichergestellt werden, dass nur ein Gerät zu einem Zeitpunkt mit ein und derselben CAN-ID sendet. Anderenfalls kann es zu Übertragungsstörungen bis zum Bus-Off kommen.
CCM	CANopen Controlling Modul, SYS TEC eigene CANopen Applikationsschnittstelle
CiA	CAN in Automation e.V.
COB	CANopen Message Object
COB-ID	CANopen Message Object ID, enthält die CAN-ID und zusätzliche Informationen die CANopen benötigt (z.B. Valid-Bit)
COP	CANopen
CPU	Central Processing Unit
CRC	Cyclic Redundancy Check
EMYC	CANopen Emergency Object (siehe CiA-301)
HBC	Heartbeat Consumer (siehe CiA-301)
HBP	Heartbeat Producer (siehe CiA-301)
LSS	Layer Setting Service (siehe CiA-301)
NMT	Network Management (siehe CiA-301)
OD	Object Dictionary
PDO	Process Data Object (siehe CiA-301)
SDO	Service Data Object (siehe CiA-301)

1 Einleitung

Dieses Handbuch beschreibt das Software-Modul zur Realisierung des Flying Masters nach dem CiA-Standard 302 Part 2 /3/. Es stellt eine Ergänzung zum CANopen Manager Software Manual L-1085 /2/ dar.

Der Flying Master wird ab CANopen Version 5.56 angeboten und ist nur für den CANopen Manager verfügbar.

2 Das Prinzip des Flying Masters

Das Erweiterungsmodul Flying Master realisiert einen CANopen Manager nach dem CiA-Standard 302 mit der Möglichkeit, den NMT Modus Master oder Slave dynamisch zu Laufzeit zu bestimmen. In einem CANopen Netzwerk darf nur ein CANopen Gerät NMT Master sein. Dieses CANopen Gerät regelt die Konfiguration der Slave-Geräte, überwacht diese Slave-Geräte und ist in der Lage, diese Slave-Geräte zu starten oder zu stoppen.

Es können sich mehrere CANopen Geräte mit dem Flying Master in einem CANopen Netzwerk befinden (*siehe Abbildung 1*). Beim Start des Netzwerkes (Zuschalten durch Power-On) handeln alle Flying Master Geräte untereinander aus, welches dieser Geräte als NMT Master agiert. Diesen Prozess nennt man NMT Flying Master Negotiation. Die Flying Master, die nicht nach dem NMT Master agieren, arbeiten als NMT Slave weiter und überwachen den aktiven NMT Master über das Heartbeat Protokoll.

Fällt der aktive NMT Master aus (ein Heartbeat Timeout wird erkannt), dann wird die NMT Flying Master Negotiation neu gestartet, und damit der aktive NMT Master unter den übrigen Fyling Master Geräten bestimmt.

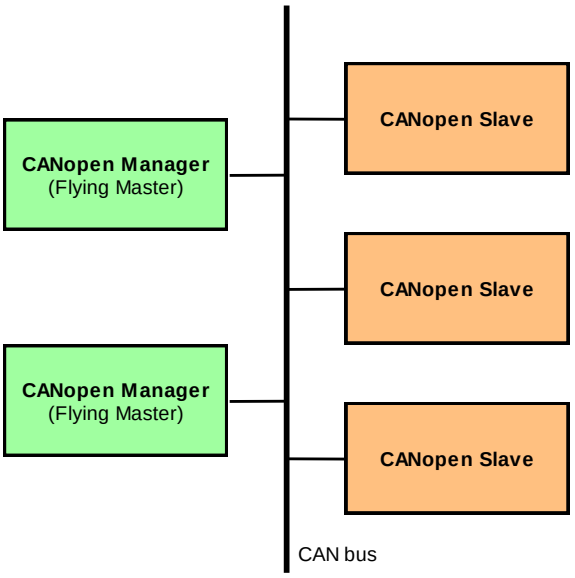

Abbildung 1: Flying Master im CANopen Netzwerk

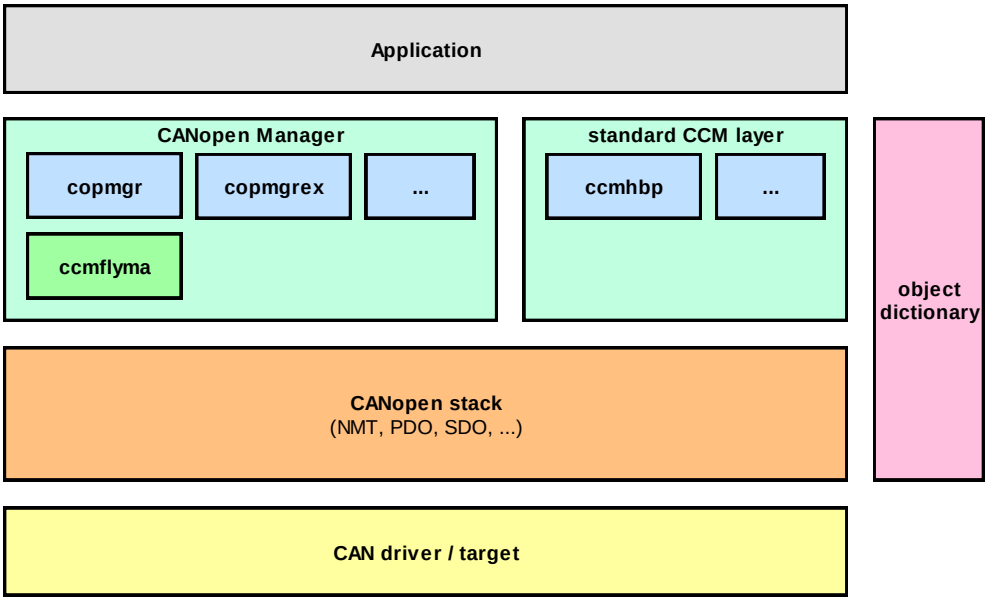

Abbildung 2: Einordnung des Flying Master in die Softwarestruktur

Funktionsprinzip *(siehe Abbildung 3)*:

(1) Wird ein CANopen Gerät gestartet, der den Flying Master beinhaltet, dann sendet dieses Gerät nach dem *NMT Master Negotiation Time Delay* den Service *Active NMT Master Detection*.

Request Service Active NMT Master Detection

CAN-ID	DLC
0x073	0

(2) Wenn sich ein weiterer Flying Master im CANopen-Netzwerk befindet, der zu dieser Zeit aktiver NMT Master ist, dann wird er diesen Service beantworten. In der Antwort sendet er seinen Prioritätslevel und seine Node-ID.

Response Service Active NMT Master Detection

CAN-ID	DLC	Data0	Data1
0x071	2	Priority	NodeID

(3) Der zugeschaltete Flying Master prüft nun die empfangene Priorität gegen seine eigene. Ist seine eigene Priorität kleiner als die des aktiven NMT Masters (eigener Prioritätslevel ist größer als der des aktiven NMT Masters) oder sogar gleich, dann startet der zugeschaltete Flying Master als NMT Slave.

(4) Ist jedoch die eigene Priorität größer als der aktive NMT Master, dann sendet der zugeschaltete Flying Master den Service *Force NMT Flying Master Negotiation*. Mit dieser CAN-Nachricht weist er dem aktiven NMT Master an, den NMT Fyling Master Negotiation Prozess bei allen verfügbaren Flying Mastern im CANopen-Netzwerk neu zu starten. Anschließend wechselt die State Machine des Flying Masters zurück zu (1).

Request Force NMT Flying Master Negotiation

CAN-ID	DLC
0x076	0

(5) Befindet sich nach Punkt (1) kein aktiver NMT Master im CANopen-Netzwerk, dann empfängt der zugeschaltete Flying Master keine Antwort auf den Service *Active NMT Master Detection*. Wird diese Situation direkt nach dem Power-On des Flying Master Gerätes erreicht, dann sendet dieses Gerät das NMT Kommando *Reset Communication* für alle Geräte im CANopen-Netzwerk und setzt seine eigene State Machine *(siehe Abbildung 3)* zurück zum Anfang: (1).

(6) Erreichte der Flying Master jedoch diese Situation in Punkt (5) nicht direkt nach dem Power-On, sondern erst nach einem vorigen Neustart der State Machine, dann sendet der zugeschaltete NMT Master den Service *NMT Flying Master Negotiation*.

Request NMT Flying Master Negotiation

CAN-ID	DLC
0x072	0

Alle Flying Master Geräte im CANopen-Netzwerk befinden sich nun in diesem Zustand (6), da zuvor alle mit dem NMT Kommando *Reset Communication* dazu gezwungen wurden, ihre State Machine neu zu starten. Der Service NMT Flying Master Negotiation wird also maximal so oft auf dem CAN-Bus erscheinen, wie viele Flying Master sich im Netzwerk befinden.

(7) Nun wartet der zugeschaltete Flying Master eine Zeit ab, die sich aus der folgenden Formel berechnet:
wait time = priority level * priority time slot + node-ID * node time slot

Die Parameter aus dieser Formel entstammen aus dem Objekt 0x1F90 (*siehe Kapitel 4.1*).

Alle Flying Master warten nun an dieser Stelle, wobei sich aber die Wartezeit bei allen Flying Master Geräten unterscheidet (wegen unterschiedliche Priorität und/oder unterschiedlicher Node-ID).

(8) Ist beim zugeschalteten Flying Master die Wartezeit abgelaufen, prüft er, ob ein anderer Flying Master auf den Service *NMT Flying Master Negotiation* geantwortet hat. Hat keiner geantwortet, dann muss der zugeschaltete Flying Master die höchste Priorität haben und sendet damit die Antwort auf den Service Service *NMT Flying Master Negotiation*. Damit startet dieser Flying Master als NMT Master.

Response NMT Flying Master Negotiation

CAN-ID	DLC	Data0	Data1
0x071	2	Priority	NodeID

(9) Hat in Punkt (8) ein anderer Flying Master vorher geantwortet, dann muss dieser eine höhere Priorität haben. Die Priorität aus der empfangenen Antwort wird mit der eigenen verglichen. Ist die eigene Priorität kleiner oder gleich, dann wird der zugeschaltete Flying Master als NMT Slave gestartet.

(10) Andernfalls handelt es sich um einen Konfigurationsfehler des gesamten Netzwerkes. Es wird in diesem Fall der Service *Force NMT Flying Master Negotiation* wie unter Punkt (4) gesendet und die State Machine neu gestartet. Unter Umständen kann sich das CANopen Netzwerk in einer immer wiederkehrenden Sequenz verfangen, da immer wieder mit dem NMT Kommando *Reset Communication* die State Machines aller Flying Master neu gestartet werden.

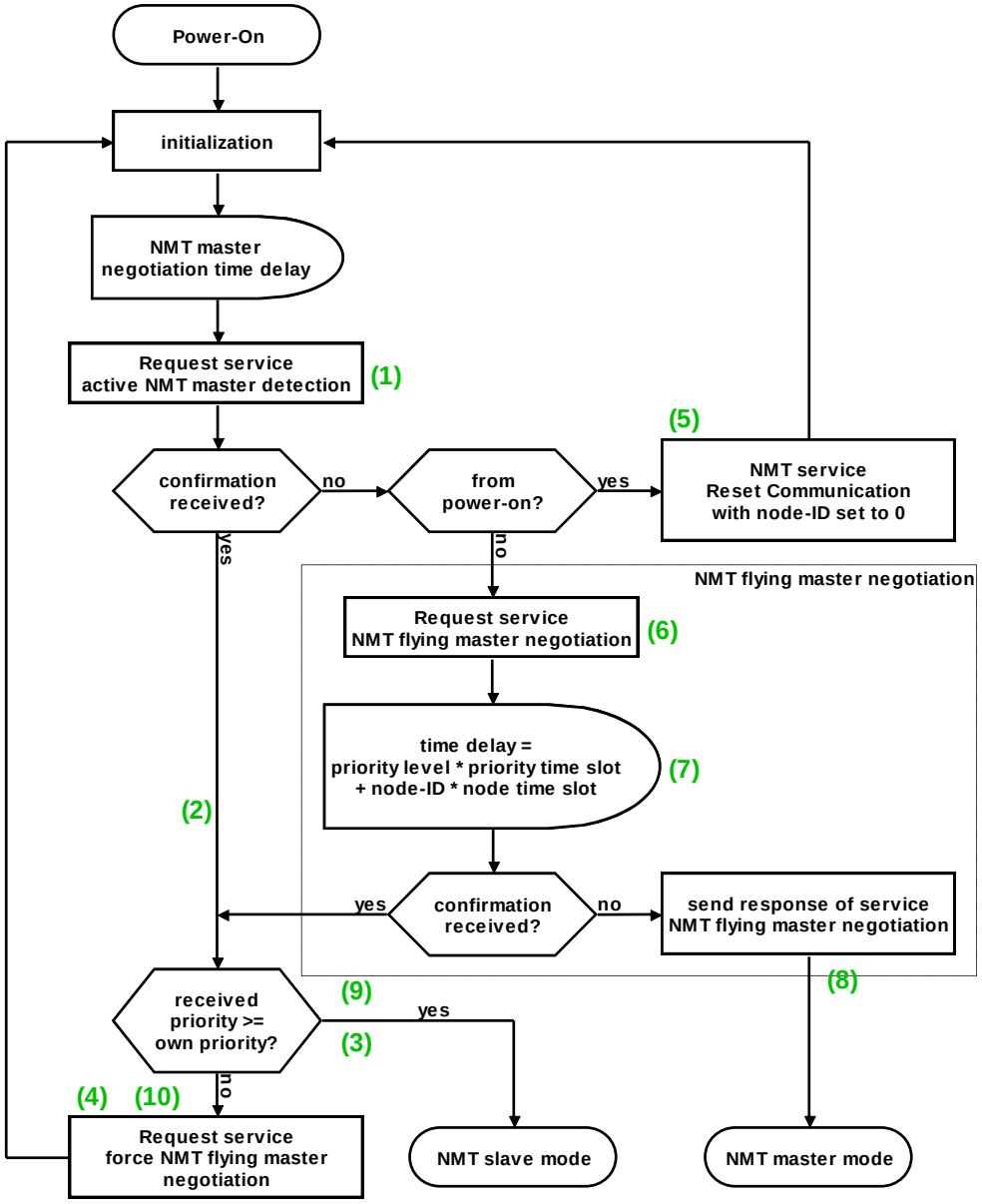

Abbildung 3: Funktionsprinzip des Flying Master

Hat der Flying Master den NMT Master Modus eingenommen, dann sendet er zyklisch den Service *NMT Flying Master Negotiation* weiter. Die Zykluszeit ist im Objekt 0x1F90 Subindex 6 konfiguriert (*siehe* Tabelle 1).

Empfängt der aktive NMT Master den Service *Force NMT Master Negotiation* von einem anderen Flying Master, dann sendet der aktive Master das NMT Kommando *Reset Communication*, um den NMT Flying Master Negotiation Prozess bei allen Geräten neu zu starten.

Achtung:

Die Applikation muss die Funktion ***CopMgrStartBootUp()*** aufrufen (siehe /2/ im Kapitel „Function CopMgrStartBootUp()“), um den NMT Flying Master Negotiation Prozess letztendlich zu starten. Der CANopen Manager verweilt im NMT State *Initialization*, solange der NMT Flying Master Negotiation Prozess andauert (d.h. bis festgelegt wurde, ob die Software im NMT Slave oder Master Modus gestartet wird – *siehe Kapitel 4.2*). Vor dem Aufruf der Funktion ***CopMgrStartBootUp()*** hat die Applikation die Möglichkeit, mit Aufruf der Funktion ***CopMgrWriteLocalObject()*** die Objekte des CANopen Managers bzw. des Flying Masters anzupassen. In *Abbildung 4* ist der prinzipielle Ablauf aus Sicht der Applikation dargestellt.

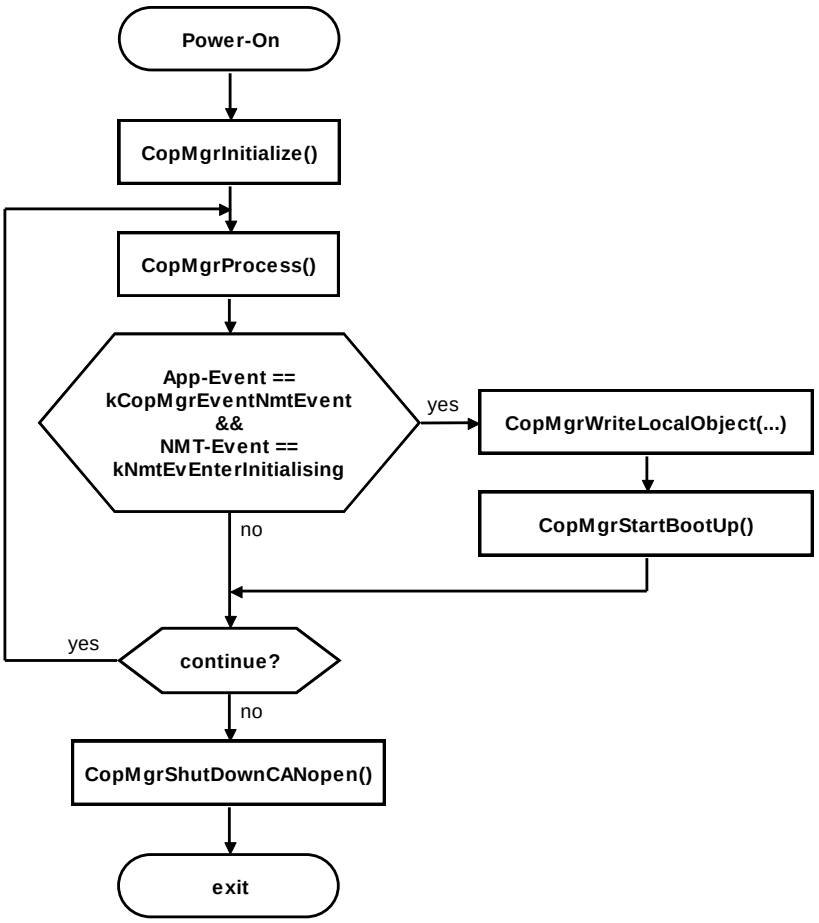

Abbildung 4: prinzipieller Ablauf in der Applikation

Es gibt keine erweiterten API-Funktionen für den Flying Master, die für die Applikation bestimmt sind. Alle internen Funktion werden ausschließlich vom CANopen Manager (Modul ***copmgr.c***) aufgerufen. Es gibt allerdings erweiterte Ereignisse für die Event-Callback Funktion in der Applikation. Diese Ereignisse werden im *Kapitel* *4.2* beschrieben.

3 Installation des Flying Masters

Das Software-Modul für das Flying Master wird über die Installation des CANopen Manager Source Codes (SO-1063) mitgeliefert. Setzen Sie bei der Installation des CANopen Manager Source Codes die Komponente für die Installation der Erweiterung für den Flying Master (SO-1114).

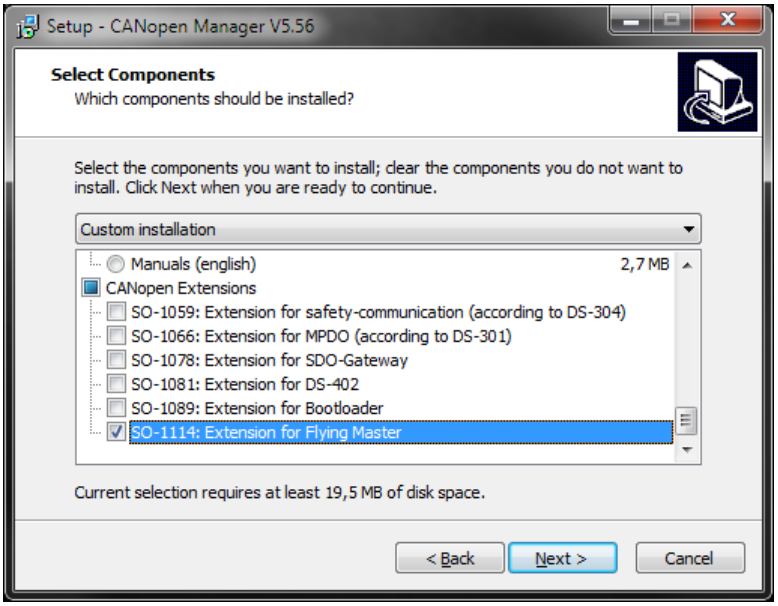

Abbildung 5: Installation des Flying Masters als Erweiterungsmodul

Während der Installation werden Sie nach dem Lizenzschlüssel für dieses Erweiterungsmodul gefragt (*siehe* *Abbildung 6*). Diesen Lizenzschlüssel erhalten Sie von der Firma SYS TEC electronic GmbH nach Bestellung und Bezahlung von SO-1114. War die Eingabe des Lizenzschlüssels korrekt, dann wird dieses Software-Modul zusätzlich zum CANopen Manager Source Code in die gleiche Verzeichnisstruktur installiert.

Hatten Sie bereits den CANopen Manager SO-1063 (ab Version V5.56) installiert und Sie haben nun das Erweiterungsmodul SO-1114 nachbestellt, dann können Sie dieses Erweiterungsmodul mit dem Aufruf von „C:\SYSTEC\COP.%Version%\Extension\SO-1114.exe“ nachinstallieren.

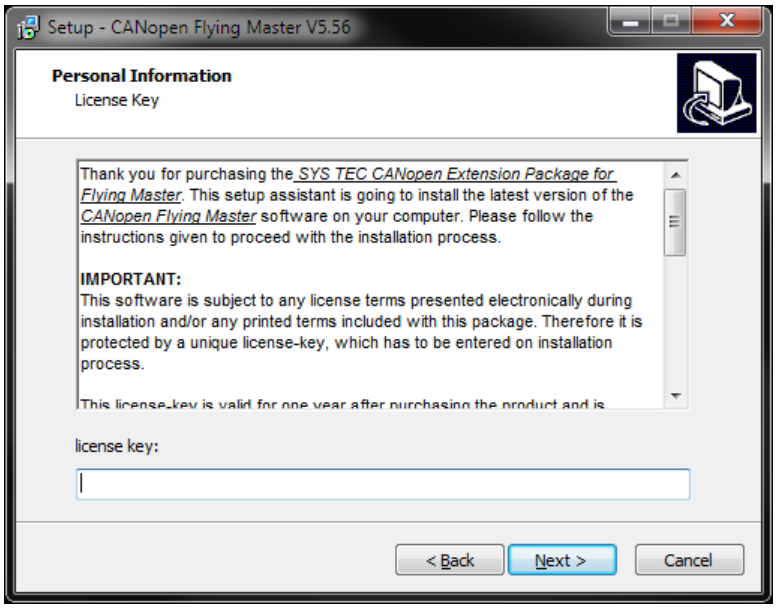

Abbildung 6: Eingabe des Lizenzschlüssels für den Flying Master

Für die Installation unter dem Linux Betriebssystem benötigen Sie einen Windows-PC. Installieren Sie hier die Software-Pakete SO-1063 und SO-1114 wie oben beschrieben. Setzen Sie dabei auch die Komponenten „ZIP: CANopen … (Unix format)“. Nach dieser Installation existieren im Verzeichnis „C:\SYSTEC\COP.%Version%.Linux“ die gepackten Dateien, die Sie auf Ihren Linux-PC transferieren und dort entpacken können.

4 Einbinden des Flying Masters

Der Flying Master besteht aus zwei Dateien: ***ccmflyma.c*** und ***ccmflyma.h***. Um den Flying Master nutzen zu können, muss die Datei ***ccmflyma.c*** dem Projekt hinzugefügt werden:

```
…\ccm\ccmflyma.c
```

Im Projekt müssen weiterhin mindestens folgende Dateien vorhanden sein (abgesehen vom CAN-Treiber, Target-Dateien, COB Modul, OBD Modul sowie CCM Module inklusive CANopen Manager Module):

```
…\copstack\nmt.c
…\copstack\nmtm.c
…\copstack\nmts.c
```

In der Konstante CCM_MODULE_INTEGRATION in der Datei ***copcfg.h*** müssen also beide Bits 2 und 3 für den NMT Master und Slave gesetzt sein. Zusätzlich müssen die Bits 12 (für den CANopen Manager) und 17 (für den Flying Master) gesetzt sein:

```
#define CCM_MODULE_INTEGRATION          (0x000210BCL)
```

Es gibt keine weiteren Konfigurationen für den Flying Master, die in der Datei ***copcfg.h*** durchgeführt werden müssen.

4.1 Objekte für den Flying Master

Im Objektverzeichnis muss das Objekt 0x1F90 nach dem CiA-Standard /3/ angelegt werden. Hier werden alle Konfigurationen des Flying Masters durchgeführt (entweder statisch als Default-Werte, von der Applikation oder von einem Konfigurationstool).

Objekt 0x1F90

Attribut	Wert
Index	0x1F90
Name	NMT flying master timing parameters
Objekttyp	ARRAY
Datentyp	UNSIGNED16
Kategorie	Optional; Mandatory for CANopen devices supporting NMT flying master
Callback	FlyMaCbConfig()
Subindex	0x00
Bedeutung	Highest sub-index supported
Zugriffsrechte	const
Wertebereich	0x06
Default-Wert	0x06
Subindex	0x01
Bedeutung	NMT master timeout
Zugriffsrechte	read-write
Wertebereich	0 – 65535
Default-Wert	100
Subindex	0x02
Bedeutung	NMT master negotiation time delay
Zugriffsrechte	read-write
Wertebereich	0 - 65535
Default-Wert	500
Subindex	0x03
Bedeutung	NMT master priority
Zugriffsrechte	read-write
Wertebereich	*Siehe Abbildung 7*
Default-Wert	Anwenderspezifisch
Subindex	0x04
Bedeutung	Priority time slot
Zugriffsrechte	read-write
Wertebereich	0 – 65535
Default-Wert	1500
Subindex	0x05
Bedeutung	CANopen device time slot
Zugriffsrechte	read-write
Wertebereich	0 – 65535
Default-Wert	10
Subindex	0x06
Bedeutung	Multiple NMT master detect cycle time
Zugriffsrechte	read-write
Wertebereich	0 – 65535
Default-Wert	4000 + 10 * Node-ID

Tabelle 1: Objekt 0x1F90 - NMT flying master timing parameters

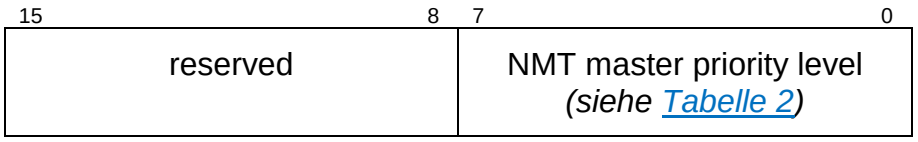

Abbildung 7: Format des Parameters NMT Master Priority

Wert	Beschreibung
0x0000	Hohe Priorität
0x0001	Mittlere Priorität
0x0002	Niedrige Priorität
0x0003	Reserviert
…	…
0x00FF	Reserviert
0x0100	Reserviert
…	…
0xFFFF	Reserviert

Tabelle 2: NMT Master Priority Level

Anlage des Objektes in der Datei ***objdict.h***:

```
OBD_BEGIN_INDEX_ROM(0x1F90, 0x07, FlyMaCbConfig)
    OBD_SUBINDEX_ROM_VAR   (0x1F90, 0x00, kObdTypUInt8,  0x01,
        tObdUnsigned8,  number_of_entries, 0x6)
    OBD_SUBINDEX_RAM_EXTVAR(0x1F90, 0x01, kObdTypUInt16, 0x83,
        tObdUnsigned16, CCM_LINK_OD_VARIABLE (OBD_LINKED_INSTANCE,
            wNmtMasterTimeout),  100)
    OBD_SUBINDEX_RAM_EXTVAR(0x1F90, 0x02, kObdTypUInt16, 0x83,
        tObdUnsigned16, CCM_LINK_OD_VARIABLE (OBD_LINKED_INSTANCE,
            wNmtNegTimeDelay),   500)
    OBD_SUBINDEX_RAM_EXTVAR(0x1F90, 0x03, kObdTypUInt16, 0x83,
        tObdUnsigned16, CCM_LINK_OD_VARIABLE (OBD_LINKED_INSTANCE,
            wNmtMasterPriority), 2)
    OBD_SUBINDEX_RAM_EXTVAR(0x1F90, 0x04, kObdTypUInt16, 0x83,
        tObdUnsigned16, CCM_LINK_OD_VARIABLE (OBD_LINKED_INSTANCE,
            wPriorityTimeSlot),  1500)
    OBD_SUBINDEX_RAM_EXTVAR(0x1F90, 0x05, kObdTypUInt16, 0x83,
        tObdUnsigned16, CCM_LINK_OD_VARIABLE (OBD_LINKED_INSTANCE,
            wCopDeviceTimeSlot), 10)
    OBD_SUBINDEX_RAM_EXTVAR(0x1F90, 0x06, kObdTypUInt16, 0x83,
        tObdUnsigned16, CCM_LINK_OD_VARIABLE (OBD_LINKED_INSTANCE,
            wMultiMasterDetectCycleTime), 4010)
OBD_END_INDEX(0x1F90)
```

Es ist wichtig, dass dieses Objekt auf die oben gezeigte Weise angelegt wird. Mit dem Makro OBD_SUBINDEX_RAM_EXTVAR() wird ein Objekt angelegt, dessen Daten nicht im Kontext des Objektverzeichnisses liegen. Das Objekt wird stattdessen auf externe Variablen ausgerichtet. Die

Adressen auf die externen Variablen werden über das Makro CCM_LINK_OD_VARIABLE() bestimmt. Sie liegen in der Instanz-Tabelle der CCM-Schicht. Die Instanz wird über die Konstante OBD_LINKED_INSTANCE (als Index) angegeben, die in der Datei ***objdict.c*** definiert ist.

Hinweis:

Der Einsatz des ODBuilders ist für den Flying Master (bzw. generell für den CANopen Manager) nicht möglich!

4.2 Erweiterte Ereignisse für die Applikation

Im Handbuch des CANopen Managers /2/ im Kapitel „Event callback function tCopMgrCbEvent()“ wird die Event-Callback Funktion für die Applikation beschrieben. Dort werden ebenfalls alle vorkommenden Ereignisse aufgelistet. Wird der Flying Master in der Konstante ***CCM_MODULE_INTEGRATION*** aktiviert, wird diese Event-Callback Funktion mit einem weiteren Ereignistyp gerufen:

```
kCopMgrEventFlyMaEvent = 0x73
```

Zu diesem neuen Ereignistyp gehört in der Union ***tCopMgrEventArg*** das Member-Element ***m_FlyMaEvent*** vom Typ ***tFlyMaEvent***. Dieser Typ ist eine Struktur mit folgendem Aufbau:

```
typedef struct _tagFlyMaEvent
{
    BYTE         m_bSubEvent;
    BYTE         m_bPriorityLevel;
    BYTE         m_bMasterNodeId;
}
tFlyMaEvent;
```

Das Member-Element ***m_bSubEvent*** enthält ein Sub-Ereignis, der das Ereignis aus dem Flying Master näher spezifiziert. In ***m_bMasterNodeId*** ist die Node-ID des aktiven NMT Masters abgelegt und in ***m_bPriorityLevel*** dessen Prioritätslevel. *Tabelle 3* listet alle möglichen Sub-Ereignisse auf.

Konstante	Wert	Bedeutung
FLYMA_EVENT_SLAVE_MODE	0x01	Die Software wurde im NMT Slave Mode gestartet auf Grund einer höheren Priorität eines anderen Flying Masters. In ***m_bPriorityLevel*** und ***m_bMasterNodeId*** sind Priorität und Node-ID des aktiven Masters angegeben.
FLYMA_EVENT_FORCE_NEG	0x02	Der Flying Master hat den Service „Force NMT Flying Master Negotiation“ gesendet, da ein anderer Flying Master eine niedrigere Priorität hat als der eigene. Die Member-Variablen ***m_bPriorityLevel*** und ***m_bMasterNodeId*** haben keine gültigen Werte.
FLYMA_EVENT_POWERON_RESET	0x03	Es hat kein CANopen Gerät auf den Service „Active NMT Master Detection“ geantwortet. Da es der erste Durchlauf ist, hat der Flying Master das NMT Kommando Reset Communication an alle CANopen Geräte gesendet. Die Member-Variablen ***m_bPriorityLevel*** und ***m_bMasterNodeId*** haben keine gültigen Werte.
FLYMA_EVENT_MASTER_MODE	0x04	Die Software wurde im NMT Master Mode gestartet, da der eigene Flying Master eine höhere Priorität hat als alle anderen im Netzwerk. Die Member-Variablen ***m_bPriorityLevel*** und ***m_bMasterNodeId*** erhalten die Priorität und die Node-ID des eigenen Knotens.
FLYMA_EVENT_ERROR_CFG	0x05	Es wurde eine Response auf den Service „NMT Flying Master Negotiation“ empfangen, obwohl die Priorität des entfernten Flying Masters kleiner ist als die eigene. Hier muss ein Konfigurationsfehler vorliegen. In ***m_bPriorityLevel*** und ***m_bMasterNodeId*** sind Priorität und Node-ID des Flying Masters angegeben, der die Response gesendet hatte.

Tabelle 3: Sub-Ereignisse für den Flying Master

Beispiel:

```
tCopKernel PUBLIC AppCbEvent (CCM_DECL_INSTANCE_HDL_
    tCopMgrEventType        EventType_p,
    tCopMgrEventArg MEM*    pEventArg_p,
    void GENERIC*           pUserArg_p)
{
  // check the event and set flags for AppProcess()
  ...
  if (EventType_p == kCopMgrEventFlyMaEvent)
  {
    // check if this node will be started as NMT slave
    if (pEventArg_p->m_FlyMaEvent.m_bSubEvent == FLYMA_EVENT_SLAVE_MODE)
    {
      // save the active NMT master node-ID for starting the heartbeat
      bActiveNmtMasterNodeId_l=pEventArg_p->m_FlyMaEvent.m_bMasterNodeId;
    }
  }
}
```

5 Ein einfaches Demo des Flying Masters

Mit dem Software-Paket SO-1114 wird eine einfache Demo mit CANopen Geräten mitgeliefert, wobei ein Gerät aus 2 Instanzen besteht. Es befinden sich also 3 logische CANopen-Geräte im CANopen-Netzwerk. Die *Abbildung 8* zeigt eine schematische Darstellung des Geräteaufbaus.

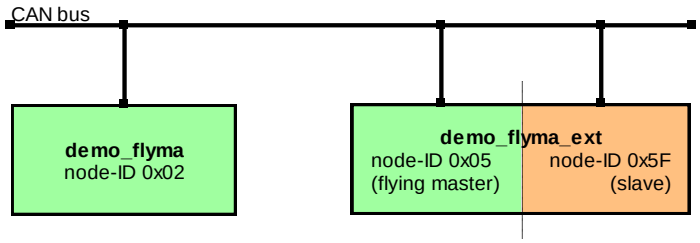

Abbildung 8: Schematische Darstellung des einfachen Demos

Das Gerät ***demo_flyma*** stellt einen einfachen CANopen Manager mit Flying Master dar. Dieser CANopen Manager hat keine weitere Funktionen implementiert (d.h. keine Knotenüberwachung, keine Konfiguration der Slaves, keine PDOs, …).

Im Gerät ***demo_flyma_ext*** werden zwei logische CANopen Geräte über zwei Instanzen realisiert. Die erste Instanz aktiviert den Flying Master über das Objekt 0x1F80 (NMT Startup – *siehe /2/*) inklusive Knotenüberwachung, automatische Konfiguration des Heartbeat bei den angeschlossenen Geräten. Die zweite Instanz ist realisiert einen CANopen Slave.

Das Demo kann wahlweise als PC-Applikation (z.B. über ein SYS TEC USB-CANmodul) oder als Hardware-Platine (z.B. STM3210C-EVAL Board) realisiert werden. Die entsprechenden Projektdateien finden Sie z.B. in folgenden Verzeichnissen:

```
…\target\stm3210c-eval\no_os\keil.uv2\demo_flyma\
…\target\stm3210c-eval\no_os\keil.uv2\demo_flyma_ext\
…\target\x86\windows\vc9\demo_flyma\
…\target\x86\windows\vc9\demo_flyma_ext\
. . .
```

Übersetzen Sie die die entsprechenden Projekte in der Debug-Konfiguration, programmieren Sie sie gegebenenfalls in die entsprechende Hardware, verbinden Sie die Hardware über ein CAN-Kabel (abgeschlossen mit den notwendigen Abschlusswiderständen von

120 Ohm an beiden Enden des CAN-Busses) und schalten Sie die Geräte ein.

Wenn Sie die Demos als PC-Applikation unter Windows mit USB-CANmodulen starten, dann benötigen Sie mindestens zwei USB-CANmodule mit der Gerätenummer 0 und ein USB-CANmodul mit der Gerätenummer 1. Statt zwei USB-CANmodule mit Gerätenummer 0 können Sie ein USB-CANmodul aber mit aktvierten USB-CANnetwork Treiber verwenden.

Über die Ausgabe-Konsole (bzw. Terminal angeschlossen an der UART des Mikrocontrollers) können Sie in den Demos ***demo_flyma*** und ***demo_flyma_ext*** beobachten, welcher der logischen Geräte den NMT Slave oder Master Mode einnimmt.

Ausgaben bei Erfolg im Demo ***demo_flyma***:

```
...
FlyMaProcess():  set state 0x0080 (kFlyMaStateReady)
CopMgrProcess(): set state 0x0001 (kCopMgrBootup)
-----------------------------------
CopMgrProcess(): set state 0x0002 (kCopMgrConfig)
0-AppCbEvent(0x61): start as master!
AppProcess(): CopMgr is started as NMT master :)
0-AppCbEvent(0x66): boot ok -> enter next state
CopMgrProcess(): set state 0x0005 (kCopMgrReady)
0-AppCbEvent(0x10): NMT-Event 0x20 -> NMT-Event 'Operational'
```

Ausgaben bei Erfolg im Demo ***demo_flyma_ext***:

```
...
1-CopMgrProcess(): set state 0x0003 (kCopMgrSlave)
1-AppCbEvent(0x60): start as slave!
1-AppProcess(): CopMgr is started as NMT slave :)
1-CopMgrProcess(): call NmtsSendBootup()
1-AppCbEvent(0x66): boot ok -> enter next state
...
0-FlyMaProcess():  send req ActMaDet
0-FlyMaProcess():  wait NMT Master Negotiation Timeout: 1000/10ms
0-FlyMaProcess():  set state 0x0002 (kFlyMaStateWaitMasterDet)
0-FlyMaProcess():  resp ActMaDet recvd (prio=1,nodeId=0x02)
0-FlyMaProcess():  enter NMT Slave Mode
0-AppCbEvent(0x73): Flying Master: slave mode (remote prio=1, node-ID=0x02)
0-FlyMaProcess():  set state 0x0080 (kFlyMaStateReady)
0-CopMgrProcess(): set state 0x0001 (kCopMgrBootup)
-----------------------------------
0-CopMgrProcess(): set state 0x0003 (kCopMgrSlave)
0-AppCbEvent(0x60): start as slave!
0-AppProcess(): CopMgr is started as NMT slave :)
0-AppProcess(): start heartbeat control for node-ID 0x02
0-CopMgrProcess(): call NmtsSendBootup()
0-AppCbEvent(0x66): boot ok -> enter next state
...
0-AppCbEvent(0x13): ErrCtrl: Node 0x02 Event=0x40 State=0x05 RequestNMT=0x05
```

Die Ausgaben variieren, je nach dem in welcher Reihenfolge die einzelnen CANopen Geräte gestartet werden. Auch die Vergabe der Prioritäten spielt eine Rolle.

In der Default-Konfiguration hat das Demo ***demo_flyma*** den Prioritätslevel 1 (mittlere Priorität – *siehe Tabelle 2*) und die erste Instanz im ***demo_flyma_ext*** den Prioritätslevel 2. Daher erlangt mit dieser Konfiguration immer das Gerät mit dem Demo ***demo_flyma*** den NMT Master Modus, auch wenn dieses Gerät erst nach dem Demo ***demo_flyma_ext*** zugeschaltet wird. Die erste Instanz im Demo ***demo_flyma_ext*** startet im NMT Slave Modus die Heartbeat Überwachung für das Gerät mit dem Demo ***demo_flyma*** (aktiver NMT Master – siehe Ausgabe „0-AppCbEvent(0x13): ErrCtrl: Node 0x02 …“ in oben dargestellter Ausgabe). Diese Heartbeat-Überwachung wird über die Applikation gestartet (Konfiguration des Heartbeat Consumers, Objekt 0x1016). Fällt der aktive Master aus, dann übernimmt automatisch die erste Instanz des Demo ***demo_flyma_ext*** den NMT Master Modus.

Werden die Demos als PC-Applikation unter Windows gestartet, dann kann den ausführbaren Dateien ein Aufrufparameter übergeben werden, der den Prioritätslevel für den Flying Master festlegt. Es sind nur Werte von 0 bis einschließlich 2 möglich. Wird ein größerer Wert übergeben, dann nimmt das Demo immer den Prioritätslevel 2 (niedrige Priorität) ein.

Beispiel mit vertauschter Priorität:

```
demo_flyma.exe  2
demo_flyma_ext.exe  1
```

Hinweis:

Im Demo ***demo_flyma*** fehlt (absichtlich) das starten der Heartbeat-Überwachung über den Heartbeat Consumer. Wird dieses Demo mit niedrigerer Priorität (d.h. mit höherem Prioritätslevel) gestartet als das Demo ***demo_flyma_ext***, dann übernimmt nicht das Demo ***demo_flyma*** den NMT Master Mode, sobald das Demo ***demo_flyma_ext*** ausfällt.

6 Hinweise für die Implementierung

- Wenn ein CANopen-Netzwerk mit mehreren Flying Master Geräten erstellt wird, dann müssen deren Konfigurationen im Objekt 0x1F90 bis auf den Prioritätslevel gleich sein. Andernfalls kann es zu Fehlverhalten im NMT Flying Master Negotiation Prozess kommen.
- Ein CANopen Manager als NMT Master hat immer bestimmte Funktionen in einem CANopen-Netzwerk. Zu diesen Funktionen zählen: Konfiguration der Slaves, Überwachung der Slaves, Starten/Stoppen des Netzwerkes usw. Ist dieser NMT Master ein Flying Master und es existieren mehrere Flying Master in Netzwerk, dann müssen die anderen Flying Master auch diese Funktionen übernehmen können, falls der aktive NMT Master ausfällt. Alle Flying Master müssen daher eine eigene Konfiguration erhalten (Knotenüberwachung, PDO-Kommunikation, usw.), die aber für das Gesamtsystem gleich ist, so dass die Hauptfunktion des gesamten CANopen-Netzwerkes gewehrleistet bleibt.
- In der Applikation muss immer die Heartbeat-Überwachung in den Flying Master Geräten für den aktiven Master gestartet werden, die selbst im NMT Slave Modus arbeiten. Es ist sogar ratsam, in jedem CANopen Gerät die Heartbeat-Überwachung für jedes andere angeschlossene Gerät zu starten (egal ob aktiver NMT Master oder Slave). Verwendet man das Modul ***cmautohb.c***, um die Heartbeat-Überwachung zu starten, dann erfolgt die automatische Aktivierung nur vom aktiven NMT Master aus. Der Flying Master, der im NMT Slave Modus gestartet wird, muss die Heartbeat-Überwachung über die Applikation starten, wie im Demo ***demo_flyma_ext*** gezeigt (siehe Funktion ***AppProcess()***, case-Zweig ***APP_STATE_MAIN_PROCESS***, Aufruf der Funktion ***CopMgrWriteLocalObject()*** für das Objekt 0x1016).

Indexverzeichnis

Dokument:	Flying Master Add-on
Dokumentnummer:	L-1491d_01, Auflage Oktober 2013

Wie würden Sie dieses Handbuch verbessern?

Haben Sie in diesem Handbuch Fehler entdeckt? Seite

Eingesandt von:

Kundennummer: _____

Name: _____

Firma: _____

Adresse: _____

Einsenden an:

SYS TEC electronic GmbH
Am Windrwad 2
D-08468 Heinsdorfergrund
GERMANY
Fax : +49 3765 38600-4100

Veröffentlicht von

SYS TEC ELECTRONIC

Best.-Nr. L-1491d_01

Printed in Germany